МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Інститут інформатики та радіоелектроніки
Факультет комп'ютерних наук та технологій
(повне найменування інституту, факультету)

Кафедра комп'ютерних систем та мереж
(повне найменування кафедри)

Пояснювальна записка

до дипломного проекту (роботи)
бакалавра
(ступінь вищої освіти (освітній ступінь))

на тему **РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ РАЙОННОГО УПРАВЛІННЯ СОЦІАЛЬНОГО ЗАХИСТУ НАСЕЛЕННЯ**

Виконав: студент 4 курсу, групи КНТ-517
спеціальності
123 «Комп'ютерна інженерія»
(код і найменування спеціальності)
Освітня програма (спеціалізація)
«Комп'ютерна інженерія»

Шатов Олег Вікторович
(прізвище та ініціали)
Керівник Проскурін М.П.
(прізвище та ініціали)
Рецензент Куляба-Харитонова Т.І.
(прізвище та ініціали)

м. Запоріжжя
2021 рік

Форма № 25

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування вищого навчального закладу)

Інститут, факультет інформатики та радіоелектроніки, комп'ютерних наук і технологій
Кафедра «Комп'ютерні системи та мережі»
Ступінь вищої освіти (освітній ступінь) бакалаврський
Спеціальність 123 Комп'ютерна інженерія
(код і найменування)
Освітня програма (спеціалізація) Комп'ютерна інженерія
(назва освітньої програми (спеціалізації)

ЗАТВЕРДЖУЮ
Завідувач кафедри Кудерметов Р.К.
_____ _____
"____" _____ 2021 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЕКТ (РОБОТУ) СТУДЕНТА

Шатова Олега Вікторовича
(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) Розробка інформаційної системи районного управління соціального захисту населення
керівник проекту (роботи) Проскурін Микола Петрович, к. т. н., доцент кафедри КСМ
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від "17" березня 2021 року № 81
2. Строк подання студентом проекту (роботи) 10 травня 2021 року
3. Вихідні дані до проекту (роботи) розробка інформаційної системи у середовищі « MS Access » у вигляді СУБД з можливостями її нарощування. Створення екранних форм, таблиць, звітів, запитів, їх зв`язків, ін.. Автоматизація операцій ввід/вивід у/із базу(и) даних, збереження, їх модифікація, формулювання інформації по реєстрам призвищ, ознак (стать, вік, статус, різновид забезпечення, пільги, ін.) у вигляді вхідних/вихідних форм, звітів, статистичних даних. Передбачити наявність методів (атрибутів/протоколів) зв`язку з головними міськими, обласними органами влади і саморядування.
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1) Аналітичний огляд;
2) Розробка бази даних та додатку;
3) Тестування розробленого додатку.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
Концептуальна модель даних системи;
Схема даних бази даних додатку;
Функціональна схема додатку;
Інтерфейс користувача додатку.

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-3	Проскурін М.П., к.т.н., доцент	17.02	17.02
нормоконтроль	Щербак Н.В., ст. викл.		

7. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (магістерської роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Визначення теми дипломного проекту та аналіз технічного завдання;	15.02.2021 р.	
2	Вивчення спеціальної літератури та аналіз загальних питань з теми дипломного проекту;	22.02.2021 р.	
3	Робота з матеріалами, аналіз та узагальнення інформації;	01.03.2021 р.	
4	Огляд та аналіз сучасних інформаційних систем ;	10.03.2021 р.	
5	Розробка концептуальної моделі інформаційної системи ІС);	26.03.2021 р.	
6	Розробка ІС, форм, таблиць, зв`язків, ін. у середовищі «MS Access»;	04.04.2021 р.	
7	Налаштування системи та інтерфейсу користувача;	14.04.2021 р.	
8	Обробка отриманих результатів, робота над пояснювальною запискою (ПЗ), рефератом;	23.04.2021 р.	
9	Охорона праці.	27.04.2021 р.	
10	Розрахунок економічних параметрів.	30.04.2021 р.	
11	Оформлення графічної частини і презентації.	06.05.2021 р.	
12	Оформлення пояснювальної записки.	15.05.2021 р.	

Студент _____ **О.В. Шатов**
(підпис) (ініціали та прізвище)

Керівник проекту (роботи) _____ **М.П. Проскурін**
(підпис) (ініціали та прізвище)

РЕФЕРАТ

ПЗ: 88с., 30 рис., 12 лістингів, 6 додатків, 11 посилань.

БАЗА ДАНИХ, ІНФОРМАЦІЙНА КОМП'ЮТЕРНА СИСТЕМА, КОНЦЕПТУАЛЬНА МОДЕЛЬ ДАНИХ, НОРМАЛІЗАЦІЯ БАЗИ ДАНИХ, РЕЛЯЦІЙНА МОДЕЛЬ ДАНИХ, СХЕМА БАЗИ ДАНИХ, СИСТЕМА УПРАВЛІННЯ БАЗАМИ ДАНИХ.

Об'єкт розробки – інформаційна комп'ютерна система районного управління соціального захисту населення.

Метою роботи є розробка інформаційної комп'ютерної системи районного управління соціального захисту населення, проведення аналізу сучасних систем управління базами даних, проектування та дослідження розроблюваної інформаційної системи.

Проект складається з трьох розділів. В кінці кожного з розділів є короткий висновок, в якому описується, що саме зроблено на даному етапі проектування.

Перший розділ присвячено аналізу технічного завдання та огляду сучасних систем управління базами даних, що використовуються для розробки та підтримки інформаційних систем.

У другому розділі виконується проектування розроблюваної системи – розробка концептуальної моделі системи та подальше її перетворення в реляційну модель, а також створення додатку у програмі Microsoft Access.

Третій розділ присвячено тестуванню розробленої системи, перевірці на відповідність технічному завданню та ER-моделі даних інформаційної системи.

В результаті виконання дипломної роботи проведено аналіз, проектування та тестування розроблюваної інформаційної системи та розроблені рекомендації щодо її модернізації, зроблено висновки про виконання завдань на тему дипломної роботи і досягнення її цілей.

ЗМІСТ

Перелік графічного матеріалу:

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

БД	–	база даних
ПЗ	–	програмне забезпечення
СУБД	–	система управління базами даних
ТЗ	–	технічне завдання
ER-модель	–	модель «сутність-зв’язок», Entity-Relationship model
SQL	–	мова структурованих запитів, Structured Query Language
VBA	–	Visual Basic для додатків, Visual Basic for Applications
XML	–	розширювана мова розмітки, eXtensible Markup Language

ВСТУП

На сучасному етапі розвитку більшості сфер виробництва/обслуговування актуальним є питання про особливості ведення бухгалтерського і податкового обліку.

Актуальність даної проблеми обумовлена безліччю помилок або неточностей, що виникають при ручному управлінні – неграмотно оформлені документи, невірно розраховані суми - що призводить до незахищеності споживача, а також може привести до серйозних проблем в підприємстві, що надає послуги.

Ціллю даної роботи є створення моделі інформаційної системи районного управління соціального захисту населення. Для досягнення заданої цілі виділено наступні задачі:

– огляд сучасних систем управління базами даних та вибір середовища розробки;

– створення структури інформаційної системи у вигляді даних, таблиць, шаблонів, форм та зв’язків інформаційної системи;

– автоматизація процесів вводу/виводу даних в базі даних;

– створення статистики на основі отриманих даних;

– моделювання та верифікація виконання пунктів технічного завдання.

1 АНАЛІТИЧНИЙ ОГЛЯД

1.1 Аналіз ТЗ та постанова задачі

На підставі аналізу технічного завдання зроблено висновок, що інформаційна система має забезпечити:

– введення і зберігання інформації про громадян;

– введення і зберігання інформації про пільги;

– на основі отриманих даних складати рішення щодо надання та виплати пільг;

– забезпечувати зберігання історії наданих пільг;

– зберігання інформації щодо наявних у громадянина пільг;

– пошук даних про громадян, пільги;

– отримання статистичних даних щодо наданих громадянам пільг.

Для рішення цих задач необхідно розробити у додатку:

– форми для введення або пошуку та редагування інформації про громадян та їх пільги, пільг;

– форму для введення або пошуку та редагування інформації про виплачені пільги: дата отримання, інформація щодо розміру виплати, залишку терміну дії пільги;

– звіт про розмір виплачених пільг за місяць;

– звіт про виплати пільг громадяну протягом всього часу;

– звіт про відносну статистику виплати пільг в залежності від місця проживання / віку громадяннина.

1.2 Огляд сучасних СУБД

Інформацією, що зберігається в базі даних (БД), може бути все що завгодно: каталог продукції, інформація про клієнтів, контент веб-сайту та ін. Для забезпечення доступу до інформації, що зберігається в базі даних, а також для управління нею, застосовують систему управління базами даних (СУБД).

СУБД - це комплекс язикових і програмних засобів, призначений для створення, ведення і сумісного використання БД багатьма користувачами. Зазвичай СУБД розрізняють по використовуваній моделі даних. Так, СУБД, що базуються на використанні реляційної моделі даних, називають реляційними СУБД. Системи управління базами даних допомагають впорядкувати інформацію, а також зв'язати бази даних між собою, при цьому надавши звіт про зміни і зареєстровані події.

Незважаючи на те, що всі системи управління базами даних виконують одну і ту ж основну задачу (тобто дають можливість користувачам створювати, редагувати і отримувати доступ до інформації, що зберігається в базах даних), сам процес виконання цього завдання варіюється в широких межах. Крім того, функції і можливості кожної СУБД можуть істотно відрізнятися.

При порівнянні різних популярних баз даних, слід враховувати зручність для користувача і масштабованість даної конкретна СУБД, а також переконатися, що вона буде добре інтегруватися з іншими продуктами, які вже використовуються. Крім того, під час вибору слід взяти до уваги вартість системи і підтримки, що надається розробником.

Якщо мова йде про вибір СУБД для підприємства, то слід взяти до уваги можливість до нарощення СУБД разом з розвитком організації.

1.2.1 Oracle

СУБД Oracle створена однойменною організацією в кінці 70-х років. Актуальна версія призначена для хмарних середовищ і може бути розміщена на одному або декількох серверах, це дозволяє управляти базами даних, які містять

мільярди записів. Деякі з функцій новітньої версії Oracle включають в себе grid framework і використання як фізичних, так і логічних структур.

Це означає, що фізичне управління даними не впливає на доступ до логічних структур. Крім того, безпека в цій версії доведена до найвищого рівня, тому що кожна транзакція ізольована від інших.

Переваги:

– найновітніші інновації серед усіх сучасних СУБД;

– велика надійність.

Недоліки:

– висока вартість;

– висока ресурсозалежність.

Oracle підходить для великих організацій, які працюють з величезними базами даних і різноманітними функціями.

1.2.2 MySQL

MySQL - одна з найпопулярніших баз даних для веб-додатків. Фактично, є стандартом для веб-серверів, які працюють під управлінням операційної системи Linux. MySQL - це безкоштовний пакет програм, однак нові версії виходять постійно, розширюючи функціонал і покращуючи безпеку. Існують спеціальні платні версії, призначені для комерційного використання.

Розробка та підтримка сайта MySQL здійснює корпорація Oracle, яка отримала права на торговельну марку разом з поглиненою Sun Microsystems, яка раніше придбала шведську компанію MySQL AB. Продукт поширюється як під GNU General Public License, так і під власною комерційною ліцензією.

Ця СУБД дозволяє вибирати різні движки для системи зберігання, які дозволяють змінювати функціонал інструменту і виконувати обробку даних, що зберігаються в різних типах таблиць. Гнучкість СУБД MySQL забезпечується підтримкою великої кількості типів таблиць: користувачі можуть вибрати як таблиці типу MyISAM, що підтримують повнотекстовий пошук, так і таблиці InnoDB, що підтримують транзакції на рівні окремих записів.

Більш того, СУБД MySQL поставляється із спеціальним типом таблиць EXAMPLE, що демонструє принципи створення нових типів таблиць. Завдяки відкритій архітектурі і GPL-ліцензуванню, в СУБД MySQL постійно з'являються нові типи таблиць. Вона також має простий у використанні інтерфейс, і пакетні команди, які дозволяють зручно обробляти величезні обсяги даних.

Переваги:

– розповсюджується безкоштовно;

– гарна технічна документація;

– велика кількість функцій навіть;

– пакет MySQL включений в стандартні репозиторії найбільш поширених дистрибутивів операційної системи Linux;

– підтримує набір призначених для користувача інтерфейсів;

– може працювати з іншими базами даних, включаючи DB2 і Oracle;

– висока надійність;

– не ресурсовитратна.

Недоліки:

– складність створення простих функцій, які в інших СУБД можуть створюватися автоматично. Наприклад, створення інкрементних резервних копій;

– відсутня вбудована підтримка XML або OLAP;

– для безкоштовної версії доступна тільки платна підтримка;

1.2.3 Microsoft SQL сервер

Це система управління базами даних, движок якої працює на хмарних серверах, а також локальних серверах, причому можна комбінувати типи застосовуваних серверів одночасно.

Однією з унікальних особливостей версії 2016 року є temporal data support (тимчасова підтримка даних), яка дозволяє відстежувати зміни даних з плином часу. Остання версія Microsoft SQL-сервер підтримує dynamic data masking (динамічне маскування даних), яке гарантує, що тільки авторизовані користувачі будуть бачити конфіденційні дані.

Переваги:

– простота використання;

– движок надає можливість регулювати і відслідковувати рівні продуктивності, які допомагають знизити використання ресурсів;

– доступ до візуалізації на мобільних пристроях;

– гарна взаємодія з іншими продуктами Microsoft.

Недоліки:

– висока вартість;

– висока ресурсозалежність;

– проблеми з використанням служби інтеграції для імпорту файлів;

Microsoft SQL Server підходить для: великих організацій, які вже використовують ряд продуктів Microsoft.

1.2.4 PostgreSQL

PostgreSQL є безкоштовною СУБД, часто використовується для ведення баз даних веб-сайтів. Може бути використаний на більшості основних платформ, включаючи Linux. Добре справляється з завданнями імпорту інформації з інших типів баз даних за допомогою власного інструментарію.

Движок БД може бути розміщений в ряді середовищ, в тому числі віртуальних, фізичних і хмарних. PostgreSQL здатна до обробки великих обсягів даних і збільшення числа одночасно працюючих користувачів. Безпека була покращена завдяки підтримці DBMS_SESSION.

Переваги:

– гарна масштабованість;

– підтримка формату json;

– велика кількість визначених функцій;

– доступний ряд інтерфейсів.

Недоліки:

– погана документація;

– складність конфігурації;

– швидкість роботи може падати під час проведення пакетних операцій або виконання запитів читання;

PostgreSQL підходить, коли є необхідність вибрати свій інтерфейс і використовувати json.

1.2.5 MongoDB

MongoDB - безкоштовна база даних, яка має комерційну версію. Вона призначена для додатків, які використовують як структуровані, так і неструктуровані дані. Ядро є дуже гнучким і працює при підключенні бази даних до додатків через драйвери MongoDB. Існує широкий вибір доступних драйверів, тому легко знайти драйвер, який буде працювати з необхідною мовою програмування.

Движок MongoDB призначений для обробки різних даних, які не можна віднести до реляційних, і може добре справлятися там, де інші движки працюють повільно або безсилі.

Переваги:

– швидкість і простота у використанні;

– движок підтримує json і інші традиційні документи NoSQL;

– дані будь-якої структури можуть бути збережені / прочитані швидко і легко.

Недоліки:

– SQL не використовується в якості мови запитів;

– програма установки може зайняти багато часу.

Підходить для організацій, що працюють з різнорідними даними, які важко піддаються класифікації.

1.2.6 Microsoft Access

Microsoft Access - реляційна СУБД, розроблена корпорацією Microsoft. Відрізняється великою кількістю вбудованих зразків для створення баз даних і вбудованим графічним дизайном.

Переваги:

– вбудований графічний дизайн;

– створення інтерфейсу бази даних;

– велика кількість вбудованих шаблонів;

Недоліки:

– низька швидкість;

– слабкі засоби захисту і відновлення інформації;

– обмеження на обсяг інформації.

1.3 Вибір та обґрунтування серверної СУБД

Microsoft Access - реляційна система управління базами даних, розроблена корпорацією Microsoft. Відрізняється великою кількістю вбудованих зразків для створення баз даних. Є можливість швидкого експорту всіх пов'язаних джерел даних в Excel. В останніх версіях програми дизайн інтерфейсу можна змінювати між кольоровим і білим.

Вбудований пошук по настройкам і функціям значно спрощує роботу з програмою. Для початку роботи досить вибрати тип даних для аналізу і Microsoft Access самостійно вибудує відповідну структуру з вбудованими засобами навігації та основними командами.

1.3.1 Основні можливості Microsoft Access

– Широкий набір інструментів для створення і роботи з базами даних;

– велика кількість встановлених шаблонів;

– підтримка апаратного прискорення;

– можливість експорту даних в SQL;

– тісна інтеграція з іншими додатками офісного пакету Microsoft Office і сервісом SharePoint;

– легке перемикання між вкладками;

– легкий доступ до проектів в хмарі;

– папку для збереження матеріалів на комп'ютері можна вибирати самостійно;

– використання макросів;

– автоматичне збереження всіх внесених змін, в тому числі і в разі екстреного закриття проекту.

В додаток вбудовано засіб перевірки правопису, який може працювати як в хмарі, так і локально. Фільтрувати і сортувати матеріал можна в автоматичному режимі з урахуванням виставлених налаштувань. Збір і відправка даних по електронній пошті можливі з інтерфейсу програми.

Технологія IntelliSense дозволяє майже повністю автоматизувати процеси, що вимагають від користувача виконання складних дій - написання коду і складних виразів. Візуальне відображення звітів підтримує форматування гістограм, завдяки чому звіти виглядають набагато більш інтуїтивно зрозумілими.

1.3.2 Елементи бази даних

1.3.2.1 Таблиці

Таблиця бази даних схожа на електронну таблицю - і там, і там інформація розташована в рядках і стовпцях. Тому імпортувати електронну таблицю в таблицю бази даних зазвичай досить легко. Основна відмінність полягає в тому, як дані структуровані.

1.3.2.2 Форми

За допомогою форм створюється призначений для користувача інтерфейс для введення і редагування даних. Форми часто містять кнопки команд і інші елементи управління, призначені для виконання різних функцій. Можна створити базу даних, не використовуючи форми, якщо просто відредагувати вже наявну інформацію в таблицях Access. Проте, більшість користувачів вважає за краще використовувати форми для перегляду, введення і редагування інформації в таблицях.

1.3.2.3 Звіти

Звіти використовуються для форматування, відомості та показу даних. Зазвичай звіт дозволяє знайти відповідь на певне питання, наприклад «Який прибуток цього року принесли нам наші клієнти?» або «В яких містах живуть наші

клієнти?» Звіти можна форматувати таким чином, щоб інформація відображалася в найбільш читабельному вигляді.

1.3.2.4 Запити

Основною функцією запитів є знаходження інформації в таблицях. Потрібна інформація зазвичай міститься в декількох таблицях, але, якщо використовувати запити, її можна переглядати в одній. Крім того, запити дають можливість фільтрувати дані (для цього задаються критерії пошуку), щоб відображалися тільки потрібні записи.

1.3.2.5 Макроси

Макроси в Access дозволяють створювати новий функціонал бази даних. Наприклад, якщо до кнопки команди в формі додати макрос, то він буде запускатися щоразу при натисканні цієї кнопки. Макроси складаються з команд, за допомогою яких виконуються певні завдання: відкриваються звіти, виконуються запити, закривається база даних і т. Д. Використовуючи макроси, можна автоматизувати більшість операцій бази даних.

1.3.2.6 Модулі

Подібно макросам, модулі - це об'єкти, за допомогою яких базу даних можна зробити більш функціональною. Але якщо макроси в Access складаються шляхом вибору зі списку макрокоманд, модулі створюються на мові Visual Basic для додатків (VBA). Модулі являють собою набори описів, інструкцій і процедур. Існують модулі класу і стандартні модулі. Модулі класу пов'язані з конкретними формами або звітами і зазвичай включають в себе процедури, які працюють тільки з цими формами або звітами. У стандартних модулях містяться загальні процедури, які пов'язані ні з яким об'єктом. Стандартні модулі, на відміну від модулів класу, перераховуються в списку Модулі в області навігації.

1.4 Висновки розділу

На даному етапі проектування визначено:

– основні завдання проектування ПЗ на основі аналізу технічного завдання, а саме: форми та звіти, призначені для пошуку, редагування та статистичної обробки інформації;

– аналізовані популярні сучасні СУБД, їх можливості, способи обробки даних, переваги та недоліки;

– аналізуючи сучасні СУБД в якості середовища розробки обрано Microsoft Access. Розглянуто основні можливості та елементи керування даними цієї СУБД.

2 РОЗРОБКА БАЗИ ДАНИХ ТА ДОДАТКУ

Базу даних можна представити на трьох рівнях: інфологічному, даталогічному та фізичному.

Мета інфологічного моделювання – забезпечення найбільш природних для людини способів збору та представлення тієї інформації, яку передбачається зберігати в базі даних. Тому інфологічну модель даних намагаються будувати по аналогії з природною мовою.

На етапі даталогічного проектування будується логічна структура БД. При цьому даталогічна модель розробляється з урахуванням конкретної реалізації СУБД основі її інфологічної моделі. Кінцевим результатом даталогічного проектування є опис структури БД на мові опису даних конкретних СУБД.

Не всі види зв’язку, що існують в інфологійній моделі наочній області, можна відобразити у даталогічній моделі. Так більшість СУБД не забезпечують підтримку зв’язку типу M:M. В цьому випадку вводиться допоміжний елемент, тобто M:N розбивається на два відношення (1:M, 1:N).

Фізична модель – прив’язка даталогічної моделі БД до середовища зберігання. Використовуються можливості даної конкретної СУБД. Приховано від розробника.

Основними конструктивними елементами інфологічної моделі є сутність, зв’язки між ними та їх властивості (атрибути).

Сутність – будь-який помітний об’єкт (об’єкт, який можна відрізнити від іншого), інформацію про який необхідно зберігати в базі даних. Сутністю можуть бути люди, місця, літаки, рейси, смак, колір і т.д. Необхідно розрізняти такі поняття, як тип сутності та екземпляр сутності. Тип сутності відноситься до набору однорідних осіб, предметів або подій, які виступаючих як ціле. Екземпляр суті відноситься до конкретної речі в наборі. Наприклад, типом сутності може бути МІСТО, а екземпляром – Москва, Київ і т.д.

Атрибут – пойменована характеристика сутності. Його найменування повинне бути унікальним для конкретного типа сутностів, але може бути однаковим для різного типа сутності (наприклад, КОЛІР може бути визначений для багатьох суті: СОБАКА, АВТОМОБІЛЬ, ДИМ). Атрибути використовуються для визначення того, яка інформація повинна бути зібрана про сутність. Прикладами атрибутів для суті АВТОМОБІЛЬ є ТИП, МАРКА, НОМЕРНИЙ ЗНАК, КОЛІР і т.д. Тут також існує відмінність між типом і екземпляром. Тип атрибуту КОЛІР має багато екземплярів або значень: Червоний, Синій, Банановий і т.д., проте кожному екземпляру суті привласнюється тільки одне значення атрибуту.

Ключ – мінімальний набір атрибутів, по значеннях яких можна однозначно знайти необхідний екземпляр сутності. Мінімальність означає, що виключення з набору будь-якого атрибуту не дозволяє ідентифікувати сутність по тих, що залишилися.

Зв’язок – асоціювання двох або більш сутностей. Якби призначенням бази даних було тільки зберігання окремих, не зв’язаних між собою даних, то її структура могла б бути дуже простою. Проте одна з основних вимог до організації реляційної бази даних – це забезпечення можливості відшукання однієї сутності за значеннями інших, для чого необхідно встановити між ними певні зв’язки. А

оскільки в реальних базах даних нерідко містяться сотні сутностей, то теоретично між ними може бути встановлено більше тисячі зв’язків. Наявність такої безлічі зв’язків і визначає складність інфологічної моделі.

2.1 Розробка інфологічної моделі предметної області

Концептуальне (інфологічне) проектування — побудова семантичної моделі предметної області, тобто інформаційної моделі найбільш високого рівня абстракції. Така модель створюється без орієнтації на якусь конкретну СУБД 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.

Найчастіше концептуальна модель бази даних включає в себе:

– опис інформаційних об'єктів або понять предметної області та зв'язків між ними;

– опис обмежень цілісності, тобто вимог до допустимих значень даних і до зв'язків між ними.

2.1.1 Модель «сутність-зв'язок»

Модель «сутність-зв'язок» (ER-модель) (англ. Entity-relationship model або англ. entity-relationship diagram) — модель даних, яка дозволяє описувати концептуальні схеми за допомогою узагальнених конструкцій блоків. ER-модель — це мета-модель даних, тобто засіб опису моделей даних. Існує ряд моделей для представлення знань, але одним з найзручніших інструментів уніфікованого представлення даних, незалежного від програмного забезпечення, що його реалізує, є модель «сутність-зв'язок». Важливим є той факт, що з моделі «сутність-

зв'язок» можуть бути породжені всі наявні моделі даних (ієрархічна, мережева, реляційна, об'єктна), тому вона є найзагальнішою.

Основні переваги ER-моделей:

– наочність;

– моделі дозволяють проєктувати бази даних з великою кількістю об'єктів і атрибутів;

ER-моделі реалізовані в багатьох системах автоматизованого проєктування баз даних;

Основні елементи ER-моделей:

– об'єкти (сутності);

– атрибути об'єктів;

– зв'язки між об'єктами.

2.1.2 Побудова інфологічної моделі

Аналізуючи технічне завдання та предметну область додатку виділяємо такі сутності: Громадянин – особа, яка отримує пільги; Пільги – описують причину та розмір пільг, що надаються, Виплата – описує спосіб надання пільги особі, якій вони надаються. Журнал виплат – служить для збереження історії виплати пільг.

2.1.2.1 Сутність Громадянин

Сутність Громадянин має поля:

– Дані свідоцтва про народження - прізвище ім’я по батькові, дата народження, серія та номер свідоцтва;

– Реєстрація місця проживання - адреса проживання громадянина;

– Дані паспорта - прізвище ім’я по батькові, дата народження, адреса проживання, вид паспорта, серія та номер паспорта;

– Ідентифікаційний код - реєстраційний номер облікової картки платника податків;

– Телефон - номер мобільного телефону;

– Дані працевлаштування - посада, адреса працевлаштування та заробітня плата;

– Сімейний дохід - дані про кількість членів сім’ї, що приносять дохід та сумарний дохід сім’ї.

2.1.2.2 Сутність Пільги

Сутність Пільги має поля:

– Законодавство - описує належність особи до певної верстви населення, яким надаються пільги: учасники бойових дій, особи з інвалідністю внаслідок війни 1, 2 та 3 груп, учасники ліквідації аварії на Чорнобильській АЕС, учасники Операції Об’єднаних сил, особи з особливими трудовими заслугами, багатодітна сім’я, ветерани військової служби, працівник служби цивільного захисту на пенсії, військовослужбовець СБУ на пенсії, депортовані особи, що досягли пенсійного віку або є інвалідами, діти-сироти та діти, що залишилися без піклування батьків;

– Розмір - відсотковий розмір пільги відносно до розміру загальної ПСП (податкової соціальної пільги).

2.1.2.3 Сутність Виплата

Сутність Виплата має поля:

– Форма - готівкова або безготівкова форма виплати;

– Періодичність - одноразові або щомісячні, щоквартальні, щорічні та інші види виплат;

– Термін - термін дії виплати, що може бути постійним або тимчасовим.

2.1.2.4 Сутність Журнал виплат

Сутність Журнал виплат має єдине поле Дата - рік, місяць, день, година, хвилина в момент якої була надана виплата.

Згідно виділених сутностей та їх атрибутам представимо інфологічну модель у вигляді ER-моделі на рисунку 2.1.

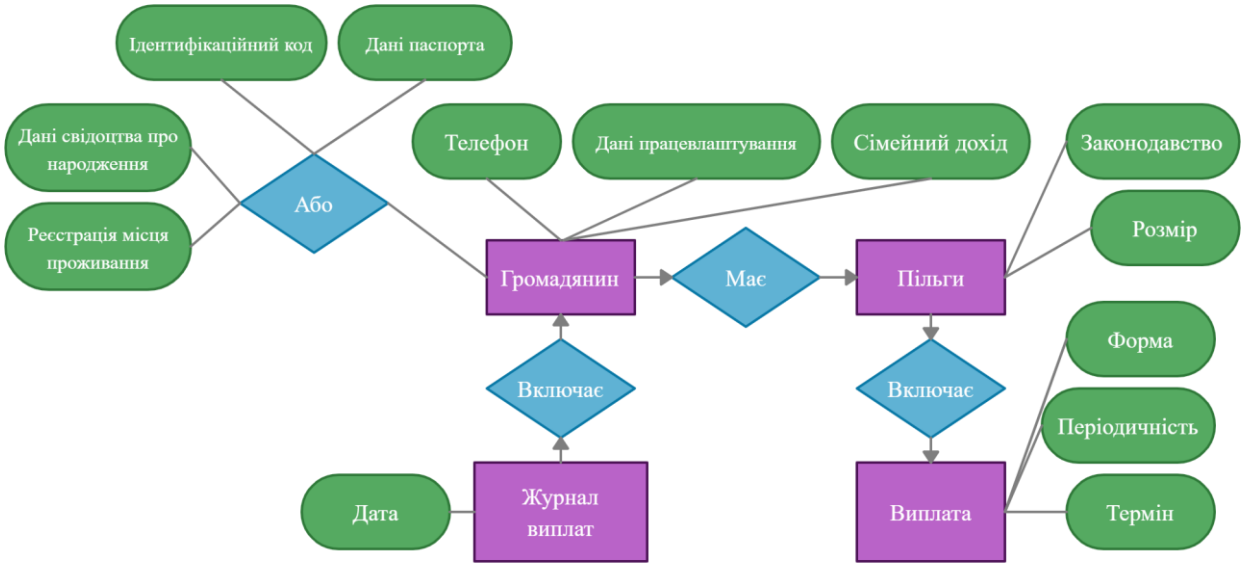


Рисунок 2.1 – ER-модель інформаційної системи

2.2 Перетворення інфологічної моделі в логічну модель

Логічна модель даних — модель даних конкретної предметної області, виражена незалежно від конкретної СУБД або технології зберігання даних, але в термінах структур даних.

У класичній теорії баз даних, модель даних - формальна теорія подання та обробки даних в СУБД, яка включає, щонайменше, три аспекти:

– аспект структури: методи опису типів і логічних структур даних в базі даних;

– аспект маніпуляції: методи маніпулювання даними;

– аспект цілісності: методи опису і підтримки цілісності бази даних.

Аспект структури визначає, що з себе логічно представляє база даних. Аспект маніпуляції визначає способи переходу між станами бази даних (тобто способи модифікації даних) і способи отримання даних з бази даних. Аспект цілісності визначає засоби описів коректних станів бази даних.

2.2.1 Види логічної моделі даних

Логічна модель бази даних може бути представлена у вигляді ієрархічної, мережевої, реляційної або об’єктно-орієнтованої моделі даних.

2.2.1.1 Ієрархічна модель даних

Ієрархічна модель даних - це модель даних, де використовується уявлення бази даних у вигляді дерева (ієрархічної) структури, що складається з об'єктів (даних) різних рівнів. Між об'єктами існують зв'язки, кожен об'єкт може включати в себе кілька об'єктів нижчого рівня. Такі об'єкти перебувають у відношенні предка (об'єкт більш близький до кореня) до нащадка (об'єкт більш низького рівня), при цьому можлива ситуація, коли об'єкт-предок має кілька нащадків, тоді як об'єкт-нащадок має лише одного предка. Об'єкти, що мають загального предка, називаються близнюками.

2.2.1.2 Мережева модель даних

Мережева модель даних - логічна модель даних, що є розширенням ієрархічного підходу, сувора математична теорія, що описує структурний аспект, аспект цілісності і аспект обробки даних в мережевих базах даних. Різниця між ієрархічною моделлю даних і мережевою полягає в тому, що в ієрархічних структурах запис-нащадок повинен мати в точності одного предка, а в мережевій структурі даних у нащадка може бути будь-яке число предків.

2.2.1.3 Реляційна модель даних

Реляційна модель даних - логічна модель даних, прикладна теорія побудови баз даних, яка є додатком до завдань обробки даних таких розділів математики, як теорія множин і логіка першого порядку. В основі цієї моделі даних лежить термін **відношення** (англ. Relation).

Відношення — математична структура, що формально визначає властивості різних об'єктів і їхні взаємозв'язки.

2.2.1.4 Об'єктно-орієнтована модель даних

Об'єктно-орієнтована модель даних - модель даних, в якій дані моделюються у вигляді об'єктів, їх атрибутів, методів і класів. Об'єктно-орієнтовані бази даних

зазвичай рекомендовані для тих випадків, коли потрібна високопродуктивна обробка даних, що мають складну структуру.

2.2.2 Обґрунтування вибору реляційної моделі даних

Подання даних в реляційній моделі даних не залежить від способу їх фізичної організації, що дозволяє описати ці дані без введення додаткових структур для машинного представлення даних. Вперше була запропонована британським ученим співробітником IBM Едгаром Франком Коддом в 1970р. В даний час фактично є стандартом, на який орієнтуються сучасні комерційні СУБД.

Для розуміння подальшого викладення необхідно розглянути базові терміни, що використовуються при побудові реляційних баз даних.

Відношення – це плоска таблиця, що складається зі стовпців і рядків.

Атрибут – це пойменований стовпець відношення.

Домен – це набір припустимих значень для одного або декількох атрибутів.

Кортеж – це рядок відношення.

Ступінь відношення визначається кількістю атрибутів, яку воно містить.

Кардинальність – це кількість кортежів, яку містить відношення.

2.2.2.1 Компоненти реляційної моделі даних

Структурна частина моделі визначає, що єдиною структурою даних є нормалізоване n-арне відношення. Відношення зручно представляти у формі таблиць, де кожен рядок є кортеж, а кожен стовпець — атрибут, визначений на деякому домені. Даний неформальний підхід до поняття відношення дає більш звичну для розробників і користувачів форму представлення, де реляційна база даних подається як кінцевий набір таблиць.

Маніпуляційна частина моделі визначає два фундаментальних механізми маніпулювання даними — реляційну алгебру і реляційне числення. Основною функцією маніпуляційної частини реляційної моделі є забезпечення заходів реляційності будь-якої конкретної мови реляційних БД: мова називається реляційною, якщо вона має не меншу виразність і потужність, ніж реляційна алгебра або реляційне числення.

Цілісна частина моделі визначає вимоги цілісності сутностей і цілісності посилань. Перша вимога полягає в тому, що будь-який кортеж будь-якого відношення відмінний від будь-якого іншого кортежу цього відношення, тобто іншими словами, будь-яке відношення має володіти первинним ключем. Вимога цілісності щодо посилань, або вимога зовнішнього ключа полягає в тому, що для кожного значення зовнішнього ключа, що з'являється у відношенні, на яке веде посилання, повинен знайтися кортеж з таким же значенням первинного ключа, або значення зовнішнього ключа повинно бути невизначеним (тобто ні на що не вказувати).

2.2.3 Етапи будування реляційної моделі даних

Перетворення інфологічної моделі у реляційну здійснюється у два етапи:

а) перетворення сутностей інфологічної моделі у відношення реляційної моделі;

1) створення відношень згідно сутностям та атрибутам концептуальної моделі;

2) визначення зв’язків між таблицями. Існує 4 типи відносин між таблицями;

– "Один до одного" (О:О)**.** У цьому випадку кожному запису однієї таблиці відповідає тільки один запис іншої таблиці;

– «Один до багатьох» (О:Б)**.** Одному запису головної таблиці (master) відповідає кілька записів підпорядкованої таблиці (detail). Тобто, кожному запису, яка є первинним ключем однієї таблиці, відповідає кілька записів пов'язаної таблиці;

– «Багато до одного» (Б:О). Декільком записам головної таблиці відповідає один запис підпорядкованої таблиці;

– «Багато до багатьох» (Б:Б). В обох таблицях існує кілька взаємопов'язаних записів. Більшість СУБД (зокрема Microsoft Access) не підтримують зв'язку «багато-до-багатьох» на рівні індексів і посилальної цілісності. Вважається, що будь-який зв'язок «багато до

багатьох» можна замінити на один або більше зв'язків «один до одного»;

3) визначення основного ключа кожної таблиці: мінімальний набір атрибутів, що ідентифікує унікальний запис в таблиці;

4) визначення зовнішних ключів (набір атрибутів, що дорівнює основному ключу зовнішньої таблиці, на яку дана таблиця посилається). При зв’язку О:О первинні ключі обох таблиць однакові, отже, зовнішній ключ не визначається. При зв’язку О:Б або Б:О зовнішній ключ визначається на стороні таблиці, яка має множинний зв’язок Б («багато»).

б) нормалізація схеми бази даних.

1) нормалізація схеми бази даних — покроковий процес розбиття одного відношення відповідно до алгоритму нормалізації на декілька відношень на базі функціональних залежностей;

2) нормальна форма — властивість відношення в реляційній моделі даних, що характеризує його з точки зору надмірності, яка потенційно може призвести до логічно помилкових результатів вибірки або зміни даних. Нормальна форма визначається як сукупність вимог, яким має задовольняти відношення.

Таким чином, схема реляційної бази даних переходить у першу, другу, третю і так далі нормальні форми. Якщо відношення відповідає критеріям нормальної форми n та всіх попередніх нормальних форм, тоді вважається, що це відношення знаходиться у нормальній формі рівня n.

2.2.3.1 Нормальні форми

Перша нормальна форма (1НФ, 1NF) утворює ґрунт для структурованої схеми бази даних:

– кожна таблиця повинна мати основний ключ: мінімальний набір колонок, які ідентифікують запис;

– уникнення повторень груп (категорії даних, що можуть зустрічатись різну кількість разів в різних записах) правильно визначаючи неключові атрибути;

– атомарність - кожен атрибут повинен мати лише одне значення, а не множину значень;

– відсутність масивів та списків в будь-якому виді.

Друга нормальна форма (2НФ, 2NF) вимагає, аби дані, що зберігаються в таблицях із композитним ключем, не залежали лише від частини ключа:

– схема бази даних повинна відповідати вимогам першої нормальної форми;

– дані, що повторно з'являються в декількох рядках, виносяться в окремі таблиці.

Третя нормальна форма (3НФ, 3NF) вимагає відсутності транзитивних функціональних залежностей неключових атрибутів від ключових:

– схема бази даних повинна відповідати всім вимогам другої нормальної форми;

– будь-яке поле, що не залежить від основного ключа має виноситись в окрему таблицю.

Нормальна форма Бойса — Кодда (НФБК)

Відношення знаходиться в НФБК тоді і лише тоді, коли детермінант кожної функціональної залежності є потенційним ключем. Якщо це правило не виконується, то, щоб привести вказане відношення до НФБК, його слід розділити на два відношення шляхом двох операцій проекції на кожну функціональну залежність, детермінант якої не є потенційним ключем:

– проекція без атрибутів залежної частини такої функціональної залежності;

– проекція на всі атрибути цієї функціональної залежності.

Визначення НФБК не потребує жодних умов попередніх нормальних форм. Якщо проводити нормалізацію послідовно, то в переважній більшості випадків при досягненні 3НФ автоматично будуть задовольнятися вимоги НФБК. 3НФ не збігається з НФБК лише тоді, коли одночасно виконуються такі 3 умови:

– відношення має 2 або більше потенційних ключів;

– ці потенційні ключі складені (містять більш ніж один атрибут);

– ці потенційні ключі перекриваються, тобто мають щонайменше один спільний атрибут.

Четверта нормальна форма (4НФ, 4NF) потребує, аби в схемі баз даних не було нетривіальних багатозначних залежностей множин атрибутів від будь чого, окрім надмножини ключа-кандидата. Вважається, що таблиця знаходиться у 4НФ тоді і лише тоді, коли вона знаходиться в НФБК та багатозначні залежності є функціональними залежностями. Четверта нормальна форма усуває небажані структури даних — багатозначні залежності.

П'ята нормальна форма (5НФ, 5NF) вимагає, аби не було нетривіальних залежностей об'єднання, котрі б не витікали із обмежень ключів. Вважається, що таблиця в п'ятій нормальній формі тоді і лише тоді, коли вона знаходиться в 4НФ та кожна залежність об'єднання зумовлена її ключами-кандидатами.

Нормальна форма домен/ключ вимагає, аби в схемі не було інших обмежень окрім ключів та доменів.

Шоста нормальна форма (6НФ, 6NF). Таблиця знаходиться у 6NF, якщо вона знаходиться у 5NF та задовольняє вимозі відсутності нетривіальних залежностей.

Шоста нормальна форма вимагає, щоб таблиці були розбиті таким чином, щоб подальша декомпозиція таблиць була неможлива без втрат даних. Дана нормальна форма майже не використовується в реальних базах даних, оскільки приведення до шостої нормальної форми призводить до втрат продуктивності, а робота з такою базою даних стає складною за рахунок збільшення кількості таблиць.

Шоста нормальна форма використовується для роботи з хронологічними базами даних.

Хронологічна база даних – база даних, що може зберігати дані, які відносяться не тільки до поточного періоду часу, але й дані, що відносяться до минулих або майбутніх періодів часу.

2.2.4 Побудова реляційної моделі даних

Основною перевагою ER-моделі є її легке конвертування до однієї з логічних моделей (ієрархічна, мережева, реляційна, об’єктно-орієнтована).

2.2.4.1 Перетворення концептуальної моделі в реляційну

Перетворення здійснюється в такому порядку:

а) аналізуючи отриману модель на першому етапі проектування, побудуємо відношення (таблиці) та їх атрибути;

1) Громадянин (Дані_свідоцтва_про_народження, Реєстрація_місця_народження, Дані_паспорта, Ідентифікаційний_код, Телефон, Дані_працевлаштування).

2) Пільги (Законодавство, Розмір).

3) Виплата (Форма, Періодичність, Кількість_Днів, Коефіцієнт_Виплати, Термін).

4) Журнал_Виплат (Дата).

б) визначення зв’язків між створеними відношеннями;

Аналузуючи технічне завдання можемо сказати, що один Громадянин може мати декілька Пільг, так само як одну й ту саму Пільгу може мати декілька Громадян. Отже, відношення Громадянин-Пільги має зв’язок Б:Б. Оскільки СУБД Mircosoft Access не підтримує зв’язок типу Б:Б, необхідно розділити це відношення, використовуючи ще одне відношення в якості з’єднуючої таблиці:

– додаємо атрибут Код_Громадянина до таблиці Громадянин та атрибут Код_Пільги до таблиці Пільги;

– створюємо нове відношення Пільги_Громадян (Код_Громадянина, Код_Пільги).

Таким чином, ми отримуємо зв’язок О:Б між таблицями Громадянин – Пільги_Громадян та зв’язок Б:О між таблицями Пільги_Громадян – Пільги. При цьому зв’язок Громадянин – Пільги видаляється.

Пільги можуть виплачуватися лише одним з визначених способів, при цьому один й той самий вид Виплати може застосовуватися для декількох Пільг. Таким чином, Пільги-Виплата має зв’язок Б:О.

Один Громадянин може отримати виплату декілька разів. Журнал_Виплат-Громадянин має відношення Б:О.

Один тип Пільги може бути використаний у Журналі декілька разів. Журнал_Виплат-Пільги має відношення Б:О.

Один тип Виплати може бути використаний у Журналі декілька разів. Журнал_Виплат-Виплата має відношення Б:О.

в) визначення основних ключів відношень;

– Громадянин (🔑Код_Громадянина, Дані_свідоцтва_про_народження, Реєстрація_місця_народження, Дані_паспорта, Ідентифікаційний_код, Телефон, Дані_працевлаштування, Сімейний_дохід);

– Пільги (🔑Код_Пільги, Законодавство, Розмір);

– Пільги_Громадян (🔑Код_Громадянина, 🔑Код_Пільги);

– Виплата (🔑Форма, 🔑Періодичність, Кількість_Днів, Коефіцієнт_Виплати 🔑Термін);

– Журнал_Виплат (🔑Дата). В даний момент таблиця Журнал_Виплат має неправильний основний ключ, оскільки декілька Громадян можуть отримати виплату Пільги в один момент часу, або один Громадян може отримати виплату з декількох видів Пільг в один момент часу. Надалі ця проблема вирішується створенням зовнішніх ключів та розширенням основного ключа;

г) визначення зовнішніх ключів відношень;

Маємо відношення О:Б Громадянин-Пільги_громадян та відношення Б:О Пільги_громадян-Пільги, але зовнішній ключ в таблиці Пільги_громадян вже було визначено при декомпозиції зв’язка Б:Б Громадянин-Пільги.

Відношення Пільги-Виплата має зв’язок Б:О. В таблиці Пільги визначаємо зовнішній ключ Пільги (🔑Код_Пільги, Законодавство, Розмір, Форма, Періодичність, Термін).

Таблиця Журнал_Виплат має зв’язки Б:О з відношеннями Громадянин, Пільги, Виплата). Визначаємо зовнішні ключі для відповідних таблиць та перевизначаємо основний ключ цієї таблиці. Отримуємо відношення Журнал_Виплат (🔑Дата, 🔑Код_Громадянина, 🔑Код_Пільги, Форма, Періодичність, Термін).

В результаті побудови реляційної моделі отримуємо такі відношення:

– Громадянин (🔑 Код_Громадянина, Дані_свідоцтва_про_народження, Реєстрація_місця_народження, Дані_паспорта, Ідентифікаційний_код, Телефон, Дані_працевлаштування, Сімейний_дохід).

– Пільги (🔑 Код_Пільги, Законодавство, Розмір, Форма, Періодичність, Термін).

– Пільги_Громадян (🔑 Код_Громадянина, 🔑 Код_Пільги).

– Виплата (🔑 Форма, 🔑 Періодичність, Кількість_Днів, Коефіцієнт_Виплати 🔑 Термін).

– Журнал_Виплат (🔑 Дата, 🔑 Код_Громадянина, 🔑 Код_Пільги, Форма, Періодичність, Термін).

д) нормалізація схеми бази даних.

Нормалізація виконується покроково, починаючи з 1-ої нормальної форми. Повний процес нормалізації БД описаний у додатку А.

Внаслідок процесу нормалізації отримуємо наступні відношення:

– Громадянин (🔑 Код_Громадянина, Прізвище, Ім’я, По_Батькові, Стать, Дата_народження, Телефон, Область, Район, Місто, Вулиця, Дом, Квартира, Сукупний_дохід, Членів_домогосподарства, Дохід_на_особу);

– Працевлаштування (🔑 Код_Працевлаштування, Код_Громадянина, Посада, Область, Район, Місто, Вулиця, Дом, Заробітня_плата);

– Громадянин_Паспорт (🔑 Код_Громадянина, Вид_паспорта, Номер_паспорта, Ідентифікаційний_код);

– Громадянин_Свідоцтво (🔑 Код_Громадянина, Номер_свідоцтва);

– Пільги (🔑 Код_Пільги, Законодавство, Розмір, Код_Виплати);

– Пільги_Громадян (🔑 Код_Громадянина, 🔑 Код_Пільги);

– Виплата (Код_Виплати, 🔑 Форма, 🔑 Код_Періодичністі, 🔑 Термін);

– Періодичність_Виплати (🔑 Код_Періодичністі, Періодичність, Кількість_днів, Коефіцієнт_Виплати);

– Журнал_Виплат (🔑 Дата, Час, 🔑 Код_Громадянина, 🔑 Код_Пільги).

2.3 Розробка структури таблиць бази даних

Усі необхідні вихідні дані для створення додатку у середовищі Microsoft Access отримані в процесі створення реляційної моделі (таблиці та їх атрибути, домени атрибутів таблиць, зв’язки між таблицями, основні ключі таблиць). Для використання Microsoft Access необхідно ознайомитися з основними елементами управління даною СУБД.

2.3.1 Елементи управління Microsoft Access

2.3.1.1 Стрічка

Стрічка – це основна заміна для меню та панелей інструментів і містить основний командний інтерфейс у програмі Access. Однією з основних переваг стрічки є те, що вона об’єднує в одному місці ті завдання або точки входу, які використовуються для відображення меню, панелей інструментів, областей завдань і інших компонентів інтерфейсу користувача, які потрібно відобразити. Таким чином, у вас є одне місце, у якому потрібно шукати команди, а не безліч місць.

Під час відкриття бази даних стрічка з'явиться у верхній частині головного вікна Access, де відображаються команди активної вкладки, як показано на рисунку 2.2.

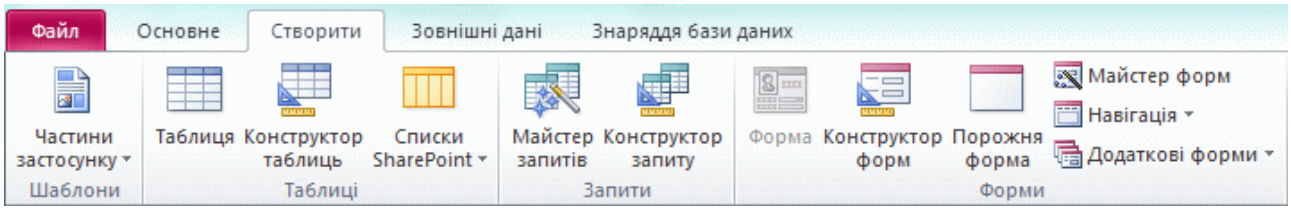


Рисунок 2.2 – Елемент управління Стрічка

Стрічка містить ряд вкладок, які містять команди. У програмі Access основні вкладки команд – це файл, основне, створення, зовнішні дані та засоби бази даних. Кожна вкладка містить групи пов'язаних команд, і ці групи мають деякі додаткові

елементи інтерфейсу користувача, наприклад, колекція, яка представляє новий тип елемента керування, який дає візуальний вибір.

Команди, доступні на стрічці, також відображають поточний активний об'єкт. Наприклад, якщо відкрито таблицю в поданні табличного подання даних, а потім на вкладці створити у групі форми натиснути кнопку форма, у програмі Access створюється форма на основі активної таблиці. Це ім'я активної таблиці вводиться у властивість RecordSource нової форми. Крім того, деякі вкладки стрічки відображаються лише в певних контекстах. Наприклад, вкладка Конструктор відображається лише під час відкриття об'єкта в режимі конструктора.

2.3.1.2 Панель швидкого доступу

Панель швидкого доступу (рисунок 2.3) – це панель інструментів, яка примикає до стрічки, яка дає змогу отримати доступ до команд одним клацанням миші. Набір команд за замовчуванням включає збереження, скасування та повторення, а також можна настроїти панель швидкого доступу, щоб додати інші команди, які часто використовуються. Ви також можете змінити розташування панелі інструментів і змінити розмір шрифту за замовчуванням для великого розміру. Поруч із вкладками команд на стрічці відображається невелика панель інструментів. Під час переходу до великого розміру панель інструментів відображається під стрічкою та подовжує всю ширину.

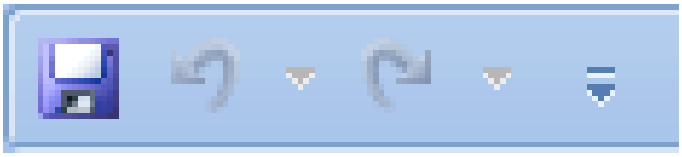


Рисунок 2.3 – Панель швидкого доступу

2.3.1.3 Область навігації

Під час відкриття бази даних або створення нової, назви об'єктів бази даних відображатимуться в області переходів. До об'єктів бази даних належать таблиці,

форми, звіти, сторінки, макроси та модулі. Область навігації показана на рисунку 2.4.

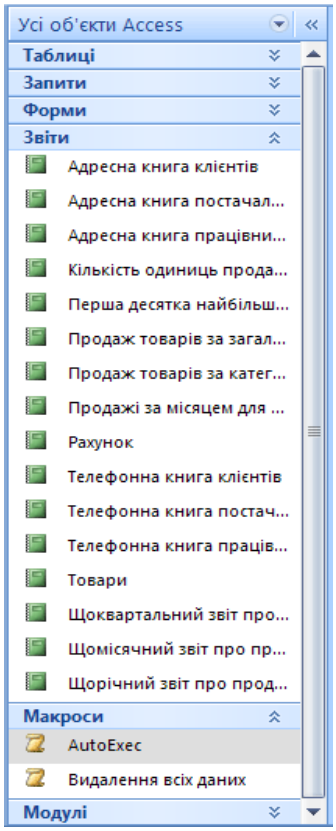


Рисунок 2.4 – Область навігації

Щоб відкрити об'єкт бази даних або використати команду для об'єкта бази даних, клацніть об'єкт правою кнопкою миші та виберіть елемент меню в контекстному меню. Команди в контекстному меню розрізняються за типом об'єкта.

2.3.1.4 Рядок стану

Рядок стану можна відобразити в нижній частині вікна. Цей стандартний елемент інтерфейсу користувача продовжує шукати повідомлення про стан, підказки властивостей, індикатори виконання тощо. У рядку стану також можна виконувати дві стандартні функції, які також відображатимуться в рядку стану інших програм Office: подання та змінення вікон і масштабування.

Ви можете швидко змінити активне вікно між одним із доступних подань, використовуючи елементи керування, доступні в рядку стану. Під час перегляду

об'єкта, який підтримує змінні масштабування, можна настроїти масштаб за допомогою повзунка в рядку стану.

Рядок стану можна ввімкнути або вимкнути в діалоговому вікні варіанти Access.

2.3.1.5 Mini-Toolbar

Ви можете легко форматувати текст за допомогою міні-панелі (рисунок 2.5). Під час виділення тексту для форматування міні-панель автоматично відображається над вибраним текстом. Якщо навести вказівник миші ближче до міні-панелі, вона зникає, і ви зможете використовувати його, щоб застосувати жирний шрифт, курсив, розмір шрифту, колір тощо. Якщо навести вказівник миші на міні-панель, вона зникає. Якщо ви не хочете використовувати міні-панель, щоб застосувати форматування тексту до виділення, просто наведіть вказівник миші на кілька пікселів, а міні-панель зникає.

Рисунок 2.5 – Mini-Toolbar

2.3.2 Створення таблиць у середовищі Microsoft Access

2.3.2.1 Конструктор таблиць

Таблиці в середовищі Microsoft Access створюються за допомогою Конструктора таблиць, що розташований у вкладці створення. Кожне поле таблиці повинне мати назву та відповідати одному з типів даних: Короткий текст, Довгий текст, Число, Дата й час, Грошова одиниця, Автонумерація, Так/Ні (логічний тип даних), Об’єкт OLE, Гіперпосилання, Вкладення, Обчислювальний, Майстер підстановок. Кожний атрибут містить необов’язкове поле опису для коротких коментарів.

Кожний атрибут таблиці має ряд властивостей, що обмежують або розширюють ввід даних в цей атрибут. Ряд властивостей залежить від типу даних атрибута. До основних властивостей поля належать: Розмір поля (кількість символів для вводу даних), Формат, Маска вводу, Значення за замовчуванням, Правило перевірки, Обов’язковість поля, Дозвіл нульової довжини, Індексація.

Ключове поле таблиці можна встановити, вибравши необхідне поле, на вкладці Конструктор натиснути на елемент Ключове поле.

2.3.2.2 Макроси

Макрос – це інструмент програми Access, за допомогою якого можна автоматизувати завдання та розширити можливості форм, звітів і елементів керування. Наприклад, коли ви додаєте кнопку до форми, ви зв’язуєте подію OnClick кнопки з макросом, що містить команди, які має виконувати кнопка щоразу після натискання.

Макроси програми Access – це ніби спрощена мова програмування, за допомогою якої ви складаєте список дій, які потрібно виконати. Під час створення макросу ви вибираєте дії з розкривного списку, а потім вводите потрібні відомості для кожної з них. Макроси допомагають розширити можливості форм, звітів і елементів керування без написання коду в модулі Visual Basic for Applications (VBA). Макроси являють собою підмножину команд, доступних у VBA, і для більшості користувачів легше побудувати макрос, ніж написати код VBA.

Наприклад, припустимо, що ви хочете запускати звіт безпосередньо з однієї із форм для вводу даних. Ви можете додати до вашої форми кнопку, а потім створити макрос, який відкриває звіт. Макрос може бути автономним (окремий об'єкт у базі даних) і прив’язаним до події OnClick кнопки або вбудованим безпосередньо в подію OnClick кнопки. У будь-якому разі, коли ви натискаєте кнопку, макрос запускається і відкриває звіт. Ці типи макросів зазвичай називають макросами інтерфейсу користувача.

Макроси даних були введені вперше в програмі Access 2010. Вони дозволяють автоматизувати завдання та додавати функції безпосередньо до

таблиць. Макроси даних та їхні дії додаються до певних подій таблиці, наприклад до події додавання до таблиці нового запису.

Макрос створюється конструктором макросів, як показано на рисунку 2.6.

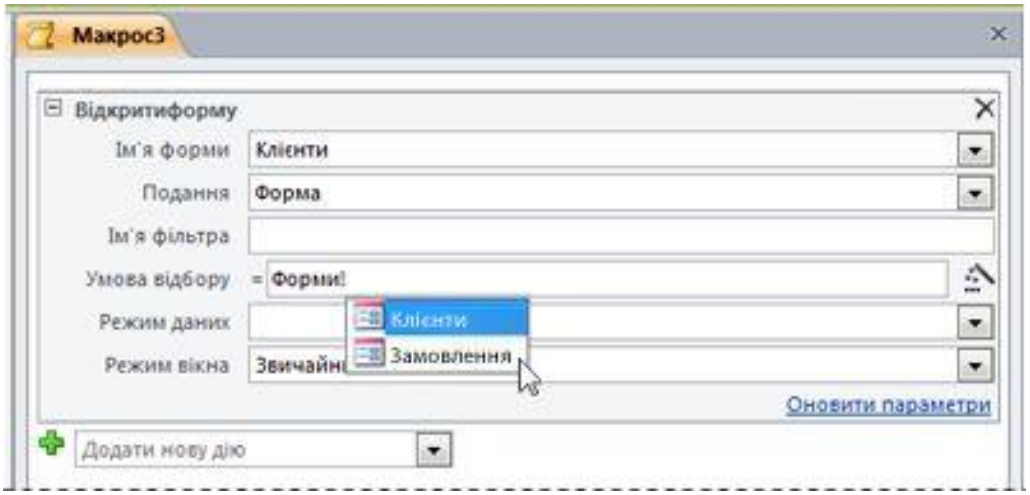


Рисунок 2.6 – Конструктор макросів

2.3.2.3 Структура таблиць бази даних

За допомогою описаних засобів для створення та налагодження таблиць СУБД Microsoft access створюємо відношення, отримані при побудові реляційної моделі даних. Повний процес створення структури таблиць, полей та функцій обробки даних описаний у додатку Б.

2.3.2.4 Схема даних

Схема даних необхідна для визначення зв’язків між таблицями. Схема даних створюється у вікні «Зв’язки» вкладки «Знаряддя бази даних» за допомогою конструктора схеми даних (рисунок 2.7).

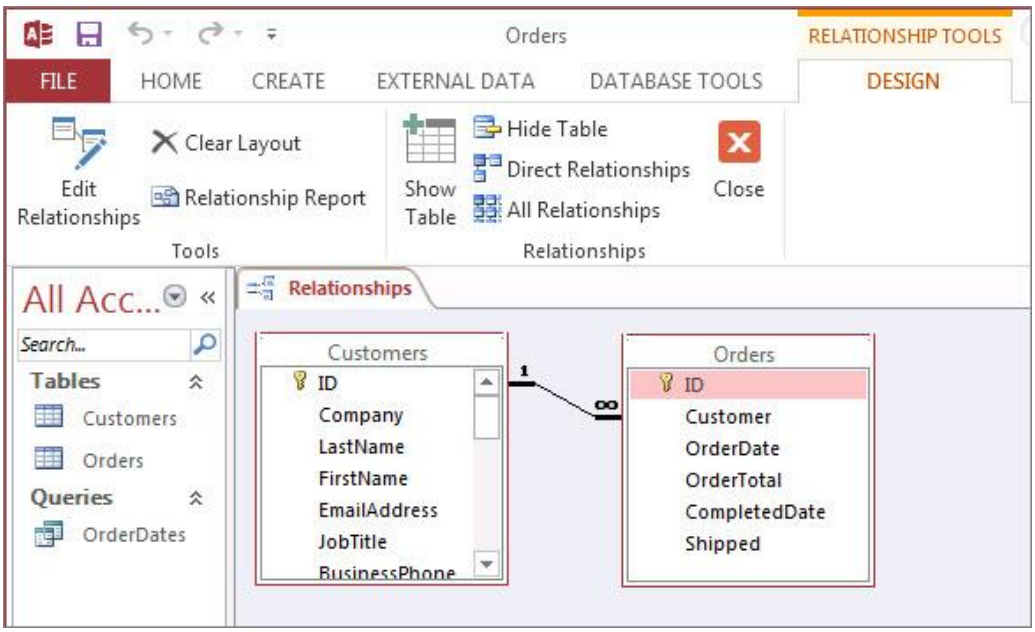


Рисунок 2.7 – Конструктор схеми даних

За допомогою визначення зв’язків між таблицями забезпечується цілісність даних в базі даних.

Щоб створити зв’язок між двома таблицями, необхідно:

– на вкладці «Конструктор» натиснути кнопку «Відобразити таблицю» та обрати необхідні таблиці;

– створити зв’язок між полями таблиць, перетягнувши поле однієї таблиці на відповідне зв’язуюче поле іншої таблиці. Якщо властивості полей обох таблиць визначені правильно, СУБД автоматично визначить тип зв’язку між таблицями.

Схема бази даних Інформаційної системи районного управління соціального захисту населення зображена.

2.4 Розробка екранних форм

Форми – це об’єкти, за допомогою яких користувачі можуть додавати, редагувати або відображати дані, що зберігаються в локальній базі даних Microsoft

Access, структура форми – це важливий аспект. Якщо очікується, що локальну базу даних Access використовуватимуть кілька користувачів, її слід правильно спроектувати, щоб вони могли ефективно й точно вводити дані.

Існує кілька способів створення форми настільної бази даних Access:

1. Створення форми з наявної таблиці/запиту за допомогою майстра форм;
2. Створення пустої форми за допомогою конструктора форм;

За допомогою майстра форм створюємо форми для редагування даних таблиць, які часто змінюються: Громадянин, Пільги, Журнал_виплат, Пільги_Громадян, Громадянин_Паспорт, Громадянин_Свідоцтво, Працевлаштування.

Дані таблиць Виплата, Періодичність_Виплати змінюються рідко, тому немає необхідності створювати форму для редагування даних у ній.

Форма Громадянин (рисунок 2.8) містить усі поля таблиці Громадянин, зв’язані підформу Працевлаштування – для додавання записів в таблицю Працевлаштування обраного громадянина та підформу Пільги – для додавання запсиів в таблицю Пільги_Громадян відповідного громадянина. Для редагування даних про паспортні дані / дані свідоцтва додано навігаційний елемент з двома вкладками: паспорт, свідоцтво. Кожна вкладка містить форму для редагування відповідної таблиці. Для навігації по записам таблиць додано кнопки переходу до наступного та попереднього запису, а також кнопку для пошуку записів за рядком. Для додавання нових записів та видалення вже існуючих записів додано відповідні кнопки «Додати запис» та «Видалити запис». Для виходу з форми додано відповідну кнопку в правому верхньому кутку.

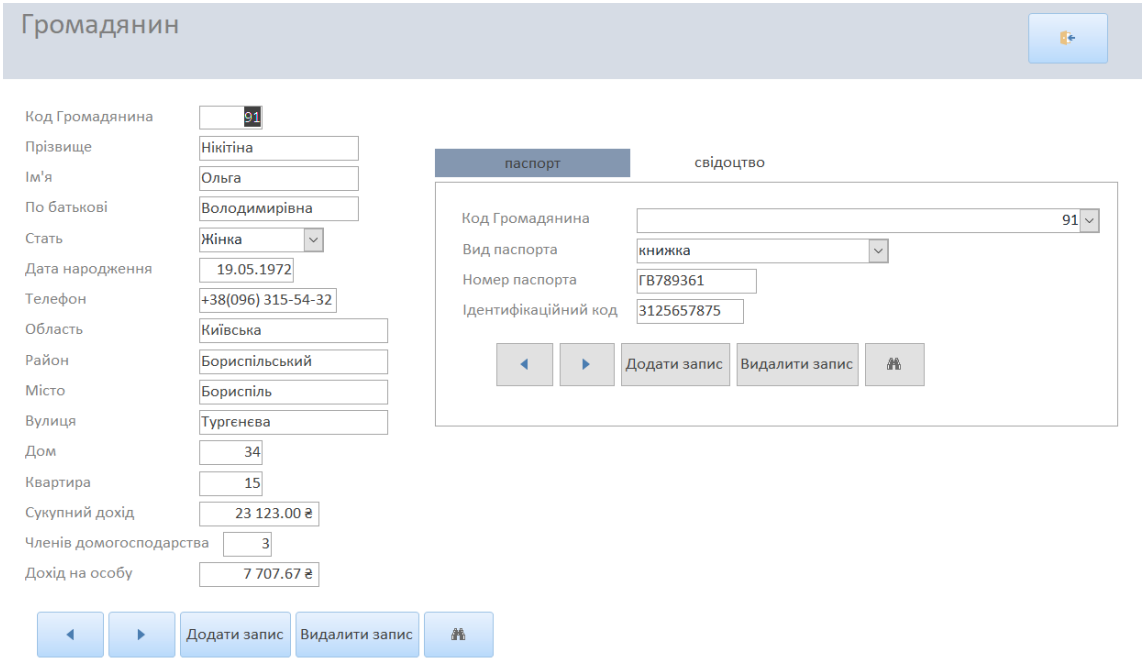


Рисунок 2.8 – Форма Громадянин

Форма Пільги (рисунок 2.9) містить усі поля таблиці Пільги та містить ті ж самі навігаційні кнопки, що й форма Громадянин.

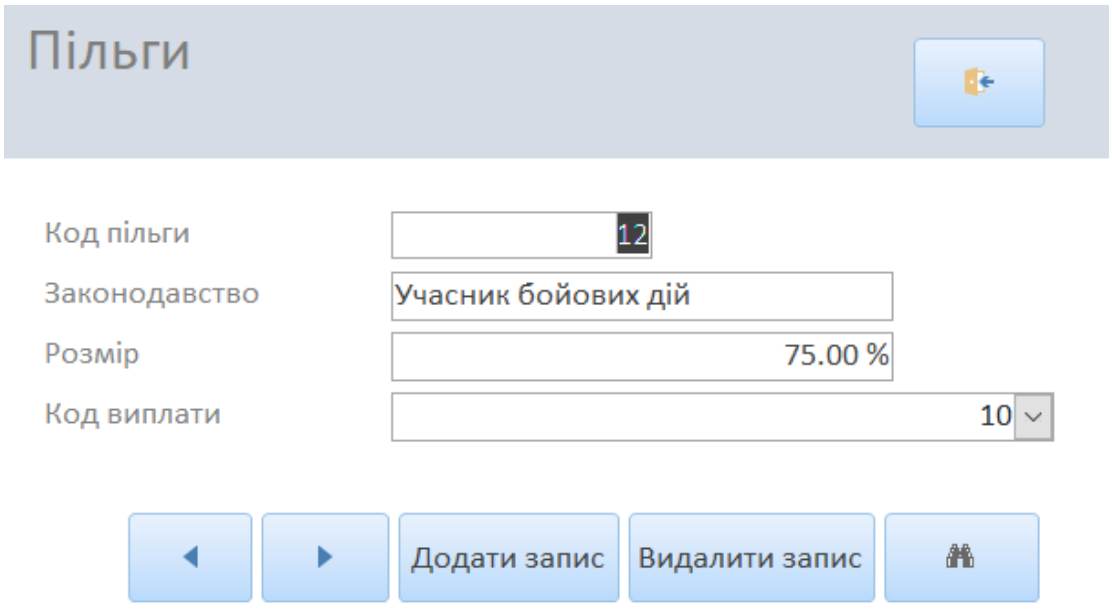


Рисунок 2.9 – Форма Пільги

Форма Журнал_Виплат (рисунок 2.10) містить усі поля таблиці Журнал_Виплат та містить ті ж самі навігаційні кнопки, що й форма Громадянин. Оскільки Громадянин не може отримати виплату з пільги, якої він не має, то в якості елементів списку поля Код_Пільги виступають лише ті, що мають відповідний запис в таблиці Пільги_Громадян.

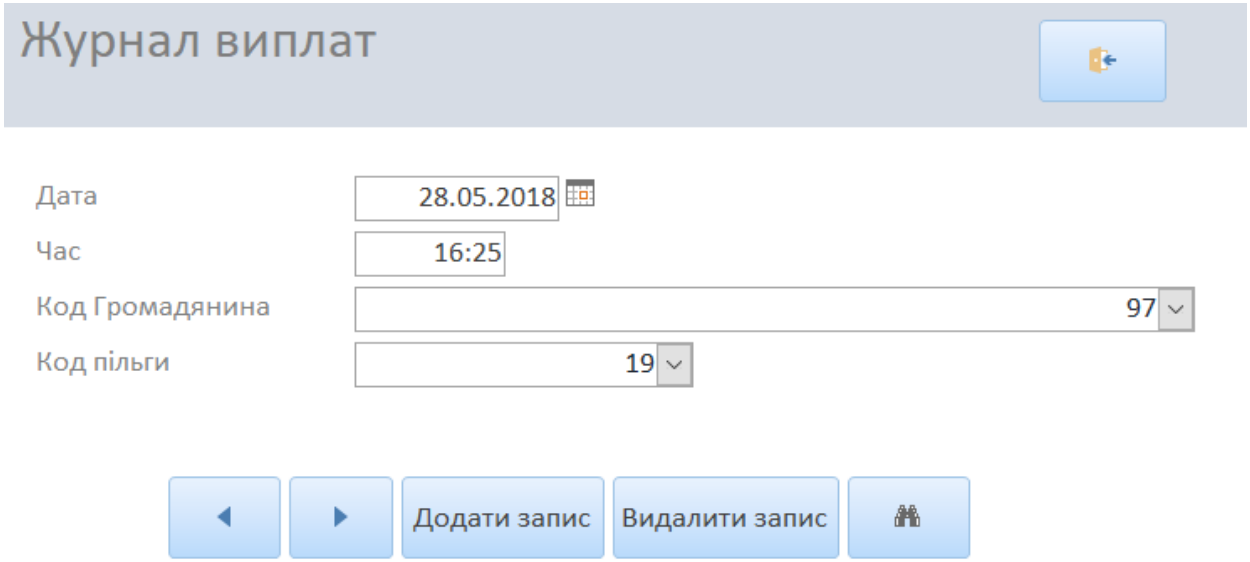


Рисунок 2.10 – Форма Журнал_Виплат

2.5 Розробка звітів

Звіти в Microsoft Access виступають в ролі основного елементу для створення статистики, форматування та підсумовування даних.

Створити звіт можна за допомогою майстра звітів на основі існуючих таблиць/запитів або створити пустий звіт за допомогою конструктора звітів.

Згідно з технічним завданням необхідно створити наступні звіти:

– звіт про розмір виплачених пільг за місяць;

– звіт про виплати пільг громадяну протягом всього часу;

– звіт про відносну статистику виплати пільг в залежності від місця проживання / віку громадяннина.

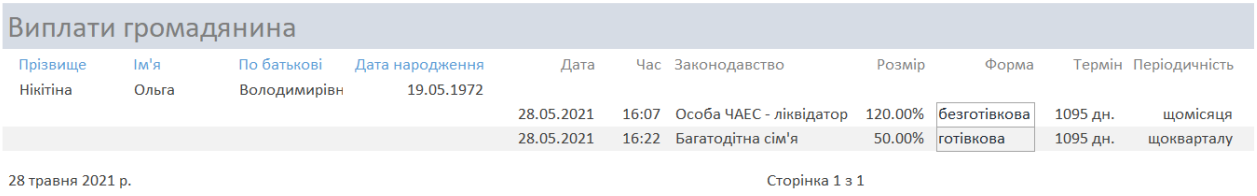
Для створення звіту про розмір виплачених за місяць пільг створюємо простий звіт за допомогою майстра звітів з використанням таблиць Журнал_Виплат, Громадянин, Пільги, Виплата, де в якості головної таблиці є Журнал_Виплат. Для перегляду даних по місяцям групуємо дані за Датою, сортування «від найновішого до найстарішого». Створюємо поле, що буде підсумовувати розмір виплачених пільг за місяць: в режимі конструктора на вкладці Конструктор обираємо елемент Підсумки – Сума. Звіт «Журнал виплат» показаний на рисунку 2.11.

Журнал виплат

Дата за Місяць	Дата	Час	Прізвище	Ім'я	По батькові	Законодавство	Розмір	Форма
травень 2021								
	06.05.2021	17:46	Тарасов	Олег	Сергійович	Трудові заслуги	75.00%	готівкова
	08.05.2021	16:25	Микитюк	Генадій	Олександрович	Учасник бойових дій	75.00%	готівкова
	08.05.2021	16:25	Шевчук	Катерина	Володимирівна	Інвалід війни 3 гр.	110.00%	готівкова
	16.05.2021	16:25	Тарасов	Олег	Сергійович	Трудові заслуги	75.00%	готівкова
	28.05.2021	15:55	Микитюк	Наталія	Романівна	Особа ЧАЕС - потерпілий	50.00%	безготівкова
	28.05.2021	16:04	Пономаренко	Роман	Валентинович	Багатодітна сім'я	50.00%	готівкова
	28.05.2021	16:07	Захарчук	Поліна	Миколаївна	Особа ЧАЕС - ліквідатор	120.00%	безготівкова
	28.05.2021	16:07	Сергієнко	Олег	Петрович	Особа ЧАЕС - ліквідатор	120.00%	безготівкова
	28.05.2021	16:07	Нікітіна	Ольга	Володимирівна	Особа ЧАЕС - ліквідатор	120.00%	безготівкова
	28.05.2021	16:17	Пономаренко	Роман	Валентинович	Особа ЧАЕС - ліквідатор	120.00%	безготівкова
	28.05.2021	16:19	Мірошниченкс	Федір	Янович	Інвалід війни 2 гр.	95.00%	безготівкова
	28.05.2021	16:20	Тарасов	Олег	Сергійович	Трудові заслуги	75.00%	готівкова
	28.05.2021	16:22	Нікітіна	Ольга	Володимирівна	Багатодітна сім'я	50.00%	готівкова
	28.05.2021	17:33	Іванов	Артур	Євгенійович	Учасник бойових дій	75.00%	готівкова
	28.05.2021	18:41	Шатов	Олег	Вікторович	Особа ЧАЕС - потерпілий	50.00%	безготівкова
						Усього	1260.00%	

Рисунок 2.11 – Звіт «Журнал виплат»

Для створення звіту про виплачені пільги конкретному громадянину створюємо параметричний запит з чотирьох параметрів, що виступають в якості потенційного ключа в таблиці Громадянин: Прізвище, Ім’я, По_Батькові, Дата_Народження. Використовуємо створений запрос в якості елементу для створення звіту в майстрі звітів. Звіт «Виплати Громадянина» показаний на рисунку 2.12.



Виплати громадянина

Прізвище	Ім'я	По батькові	Дата народження	Дата	Час	Законодавство	Розмір	Форма	Термін	Періодичність
Нікітіна	Ольга	Володимирівн	19.05.1972							
				28.05.2021	16:07	Особа ЧАЕС - ліквідатор	120.00%	безготівкова	1095 дн.	щомісяця
				28.05.2021	16:22	Багатодітна сім'я	50.00%	готівкова	1095 дн.	щокварталу

28 травня 2021 р. Сторінка 1 з 1

Рисунок 2.12 – Звіт «Виплати Громадянина»

Звіт про отримання пільг в залежності від віку громадянина створюємо за допомогою майстра звітів, де в якості елемента групування використовуємо вираз, що обчислює вік громадянина:

$$Year(Now()) - Year([\text{Дата\_народження}])$$

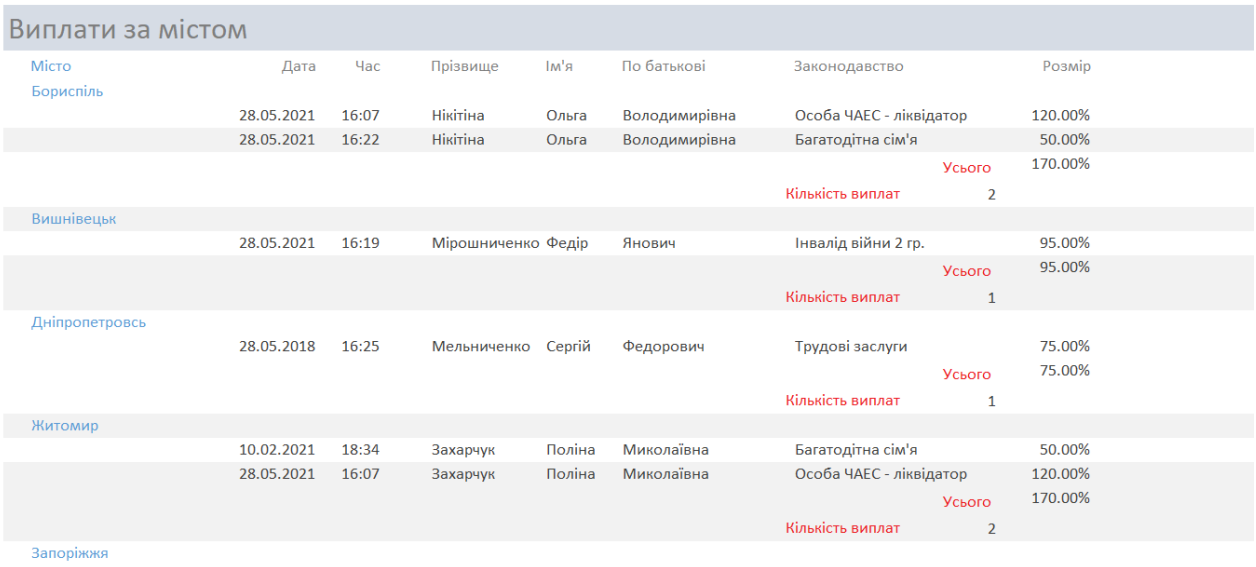
Звіт «Статистика за містом» показаний на рисунку 2.13.

Виплати за віком

Прізвище	Ім'я	По батькові	Дата народження	Дата	Час	Законодавство	Розмір
Вік 21							
Шатов	Олег	Вікторович	30.08.2000	28.05.2021	18:41	Особа ЧАЕС - потерпілий	50.00%
						Усього	50.00%
						Кількість виплат 1	
Вік 25							
Микитюк	Генадій	Олександров	01.04.1996	24.03.2021	16:54	Особа ЧАЕС - потерпілий	50.00%
Микитюк	Генадій	Олександров	01.04.1996	08.05.2021	16:25	Учасник бойових дій	75.00%
						Усього	125.00%
						Кількість виплат 2	
Вік 31							
Захарчук	Поліна	Миколаївна	14.09.1990	10.02.2021	18:34	Багатодітна сім'я	50.00%
Захарчук	Поліна	Миколаївна	14.09.1990	28.05.2021	16:07	Особа ЧАЕС - ліквідатор	120.00%
						Усього	170.00%
						Кількість виплат 2	

Рисунок 2.13 – Звіт «Статистика за віком»

Звіт про отримання пільг в залежності від міста проживання громадянина створюємо за допомогою майстра звітів, де в якості елемента групування використовуємо використовуємо поле Місто таблиці Громадянин. Звіт «Статистика за містом» показаний на рисунку 2.14.



Виплати за містом

Місто	Дата	Час	Прізвище	Ім'я	По батькові	Законодавство	Розмір
Бориспіль							
	28.05.2021	16:07	Нікітіна	Ольга	Володимирівна	Особа ЧАЕС - ліквідатор	120.00%
	28.05.2021	16:22	Нікітіна	Ольга	Володимирівна	Багатодітна сім'я	50.00%
						Усього	170.00%
						Кількість виплат 2	
Вишнівецьк							
	28.05.2021	16:19	Мірошниченко	Федір	Янович	Інвалід війни 2 гр.	95.00%
						Усього	95.00%
						Кількість виплат 1	
Дніпропетровсь							
	28.05.2018	16:25	Мельниченко	Сергій	Федорович	Трудові заслуги	75.00%
						Усього	75.00%
						Кількість виплат 1	
Житомир							
	10.02.2021	18:34	Захарчук	Поліна	Миколаївна	Багатодітна сім'я	50.00%
	28.05.2021	16:07	Захарчук	Поліна	Миколаївна	Особа ЧАЕС - ліквідатор	120.00%
						Усього	170.00%
						Кількість виплат 2	
Запоріжжя							

Рисунок 2.14 – Звіт «Статистика за містом»

2.6 Розробка екранної форми додатку

Головна кнопкова форма (рисунок 2.15) виступає в якості головного навігаційного елементу керування базою даних. На ній розміщують елементи бази даних, що використовуються найчастіше в процесі використання бази даних: форми, звіти, запити.

Починаючи з Access 2010, диспетчер кнопкових форм недоступний на стрічці, тому спочатку потрібно додати команду на панель швидкого доступу: виберіть Файл-Параметри-Панель швидкого доступу. У полі Вибрати команди зі списку виберіть пункт Усі команди. Виберіть Диспетчер кнопкових форм , а потім натисніть кнопку Додати. Натисніть кнопку ОК , щоб зберегти зміни, і закрийте діалогове вікно " настройки Access ".

Після цього елемент керування Диспетчер кнопкових форм з’явиться на панелі швидкого доступу.

За допомогою диспетчера кнопкових форм створюємо сторінки головної кнопкової форми: Форми, Звіти та елементи для переходів, що відкривають

відповідні звіти/форми. Для можливості повернення до головної сторінки додаємо відповідну кнопку повернення на кожній сторінки головної кнопкової форми.

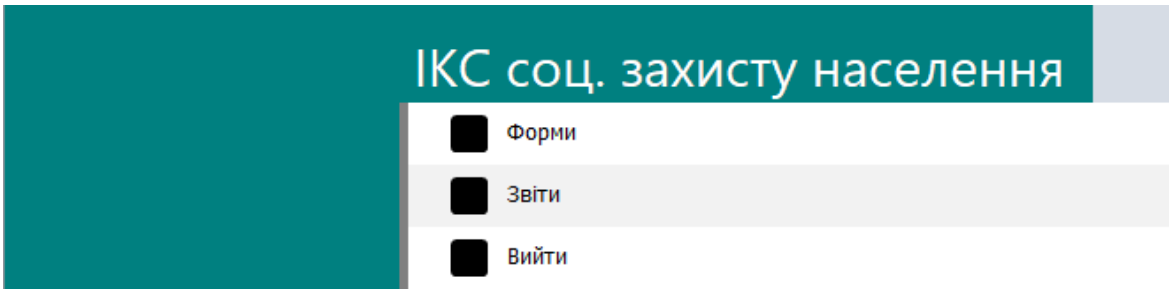


Рисунок 2.15 – Головна кнопкова форма додатку

2.7 Висновки розділу

На даному етапі проектування:

– спроектована концептуальна ER-модель бази даних;

– спроектована реляційна модель даних: визначено таблиці бази даних, їх поля та властивості полей;

– визначено зв’язки між таблицями бази даних, створено схему даних;

– створено форми для зміни, видалення та додавання записів таблиць;

– створено звіти для перегляду, підсумовування та створення статистики на основі даних в таблицях.

3 ТЕСТУВАННЯ РОЗРОБЛЕНОГО ДОДАТКУ

3.1 Системне тестування

Системне тестування є одним з рівнів тестування програмного забезпечення. Системне тестування тестує інтегровану систему для перевірки відповідності всім вимогам. Під час системного тестування перевіряються дві складові системи: база даних і додатки. Основне завдання системного тестування — у виявленні дефектів, пов’язаних з роботою системи в цілому, таких як

– невірне використання ресурсів системи,

– непередбачувані комбінації даних користувальницького рівня,

– несумісність із оточенням,

– непередбачувані сценарії використання,

– відсутня або невірна функціональність,

– незручність у застосуванні тощо.

Системне тестування відбувається над проектом у цілому за допомогою методу «чорної скриньки» (або поведінкове тестування - стратегія (метод) тестування функціональної поведінки об'єкта (програми, системи) з точки зору зовнішнього світу, при якому не використовується знання про внутрішній устрій тестованого об'єкта. Під стратегією розуміються систематичні методи відбору та створення тестів для тестового набору. Стратегія поведінкового тесту виходить з технічних вимог і їх специфікацій). Структура програми не має ніякого значення, для перевірки доступні тільки входи й виходи, видимі користувачеві. Тестуванню підлягають коди та документація користувача.

3.2 Функціональне тестування

Мета функціонального тестування (Functional Testing) – виявлення невідповідностей між реальною поведінкою реалізованих функцій і очікуваною поведінкою відповідно до специфікації і вимог. Функціональні тести повинні охоплювати всі реалізовані функції з урахуванням найбільш ймовірних типів помилок. Тестові сценарії, що поєднують окремі тести, орієнтовані на перевірку якості розв'язку функціональних задач.

До задач функціонального тестування належать:

– ідентифікація множини функціональних вимог;

– ідентифікація зовнішніх функцій і побудова послідовностей функцій відповідно до їхнього використання в програмному засобі;

– ідентифікація множини вхідних даних кожної функції і визначення областей їхньої зміни;

– побудова тестових наборів і сценаріїв тестування функцій;

– виявлення і подання усіх функціональних вимог за допомогою тестових наборів і проведення тестування помилок у програмі і при взаємодії із середовищем.

Тести, створювані за проєктною інформацією, пов'язані зі структурами даних, алгоритмами, інтерфейсами між окремими компонентами і застосовуються для тестування компонентів і їхніх інтерфейсів. Основна мета – забезпечення повноти і погодженості реалізованих функцій і інтерфейсів між ними.

3.3 Юзабіліті тестування

Юзабіліті-тестування (перевірка ергономічності, Usability Testing — тестування зручності) — дослідження, що виконується з метою визначення

зручності деякого штучного об'єкту (веб-сторінка, користувальницький інтерфейс або пристрій) для його подальшого застосування. Таким чином, перевірка ергономічності вимірює ергономічність об'єкта або системи. Перевірка ергономічності зосереджена на певному об'єкті або невеликому наборі об'єктів, у той час як дослідження взаємодії людина-комп'ютер в цілому — формулюють універсальні принципи.

Перевірка ергономічності — метод оцінки зручності продукту у використанні, заснований на залученні користувачів як тестувальників, випробувачів і підсумовуванні отриманих від них висновків.

3.4 Висновки розділу

На даному етапі проектування виконано тестування додатку трьома способами: системне, функціональне та юзабіліті тестування.

При системному тестуванні усунено: помилки форматування даних при перегляді звітів, зміна розташування елементів навігації форм.

На етапі функціонального тестування виявлено та усунено: помилку вибору невірних даних у полі Код_Пільги форми Журнал_Виплат за допомогою створення додаткового макросу, що видаляє та відновлює список даних у полі Код_Пільги при зміні даних у полі Код_Громадянина.

На етапі юзабіліти тестування налаштоване автоматичне відкриття головної кнопкової форми при відкритті додатку бази даних.

ВИСНОВКИ

В процесі розробки бази даних визначено основні завдання та цілі проектування. Розглянуто сучасні системи управління базами даних та в якості середовища розробки додатку обрано Microsoft Access.

Спроектовано концептуальну ER-модель бази даних. На базі ER-моделі спроектовано реляційну модель даних. Визначено таблиці бази даних, їх поля та зв’язки між таблицями.

Розроблено екранні форми та звіти для редагування записів бази даних та перегляду даних/статистики. Створено головну екранну форму для навігації по додатку.

Для перевірки додатку на відповідність технічному завданню проведено тестування додатку на трьох рівнях: системне тестування, функціональне тестування, юзабіліті тестування.

Результатом роботи є база даних районного управління соціального захисту населення з можливостями її розширення/інтеграції до баз даних більш високого рівня: район, область, Україна.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Порівняння характеристик сучасних СУБД. URL: https://drach.pro/blog/hi-tech/item/145-db-comparison (дата звернення: 20.05.2021).

2. Етапи проектування баз даних. URL: https://uk.wikipedia.org/wiki/Проєктування_бази_даних (дата звернення: 20.05.2021).

3. Модель «сутність – зв’язок». URL: https://uk.wikipedia.org/wiki/Модель_«сутність_—_зв%27язок» (дата звернення: 20.05.2021).

4. Види форми виплати пільг на оплату житлово-комунальних пільг. URL: https://biz.ligazakon.net/news/191022_plgi-na-oplatu-zhitlovo-komunalnikh-poslug-nadayutsya-po-novomu (дата звернення: 20.05.2021).

5. Конституція України: Закон України від 21.11.2020, документ 94-94-п, підстава 1127-2020-п // Про порядок надання пільг, передбачених Законом України. URL: https://zakon.rada.gov.ua/laws/show/94-94-%D0%BF#Text (дата звернення: 20.05.2021).

6. Інфологічна модель даних «сутність-зв’язок». URL: https://studfile.net/preview/7363846/ (дата звернення: 20.05.2021).

7. Пільги на житлово-комунальні послуги. URL: https://wiki.1551.gov.ua/pages/viewpage.action?pageId=13762669 (дата звернення: 20.05.2021).

8. Нормальні форми баз даних. URL: https://info-comp.ru/first-normal-form (дата звернення: 20.05.2021).

9. Функція VBA для нормалізації рядку. URL: http://msa.polarcom.ru/st/s0000260.htm (дата звернення: 20.05.2021).

10. Офіційна документація Microsoft Access. URL: https://support.microsoft.com/uk-ua/office (дата звернення: 20.05.2021).

11. Андріїв В.М. Класифікація соціальних пільг. Київський національний університет імені Тараса Шевченка. С.3-6.

ДОДАТОК А

Нормалізація схеми бази даних

В результаті побудови реляційної моделі отримуємо такі відношення:

– Громадянин (🔑Код_Громадянина, Дані_свідоцтва_про_народження, Реєстрація_місця_народження, Дані_паспорта, Ідентифікаційний_код, Телефон, Дані_працевлаштування, Сімейний_дохід);

– Пільги (🔑Код_Пільги, Законодавство, Розмір, Форма, Періодичність, Термін);

– Пільги_Громадян (🔑Код_Громадянина, 🔑Код_Пільги);

– Виплата (🔑Форма, 🔑Періодичність, Кількість_Днів, Коефіцієнт_Виплати 🔑Термін);

– Журнал_Виплат (🔑Дата, 🔑Код_Громадянина, 🔑Код_Пільги, Форма, Періодичність, Термін).

Необхідно провести нормалізацію схеми бази даних для уникнення аномалій бази даних: повторення груп даних; дані, що посилаються на неіснуючі поля; при видаленні інформації видаляються поля, що не зв’язані з видаленим кортежем даних та інші аномалії. Також нормалізація бази даних спрощує обробку інформації в базі даних.

Нормалізація бази даних виконується покроково слідуючи правилам нормальних форм.

Перша нормальна форма (1НФ, 1NF):

– кожна таблиця повинна мати основний ключ: мінімальний набір колонок, які ідентифікують запис;

Основні ключі вже визначені для кожної з таблиць.

– уникнення повторень груп (категорії даних, що можуть зустрічатись різну кількість разів в різних записах) правильно визначаючи неключові атрибути;

Повторення груп даних виникає в таблицях Пільги (🔑Код_Пільги, Законодавство, Розмір, Форма, Періодичність, Термін) та Журнал_Виплат (🔑Дата, 🔑Код_Громадянина, 🔑Код_Пільги, Форма, Періодичність, Термін).

Форма, Періодичність та Термін виплати залежать від виду Пільги, але не від Журналу виплат. Отже, необхідно видалити дублюючі неключові поля з таблиці Журнал_Виплат. Отримуємо відношення Журнал_Виплат (🔑Дата, 🔑Код_Громадянина, 🔑Код_Пільги).

Якщо один громадянин має декілька місць працевлаштування, то може виникнути повторення записів даних про громадянина. Виконуємо декомпозицію таблиці Громадянин на два відношення: Громадянин (🔑Код_Громадянина, Дані_свідоцтва_про_народження, Реєстрація_місця_народження, Дані_паспорта, Ідентифікаційний_код, Телефон, Дані_працевлаштування, Сімейний_дохід) та Працевлаштування (🔑Код_Працевлаштування, Код_Громадянина, Посада, Область, Район, Місто, Вулиця, Дом, Заробітня_плата). Відношення Працевлаштування має зв’язок Б:О (багато-до-одного) з таблицею Громадянин.

Атрибути Форма, Періодичність, Термін таблиці Пільги є зовнішнім ключем до таблиці Виплата. Для уникнення повторення даних створимо в таблиці Виплата унікальне числове поле Код_Виплати, яке однозначно буде ідентифікувати запис в таблиці. Таким чином в якості зовнішнього ключа в таблиці Пільги замість атрибутів Форма, Періодичність, Термін буде атрибут Код_Виплати. Отримуємо відношення: Пільги (🔑Код_Пільги, Законодавство, Розмір, Код_Виплати) та Виплата (Код_Виплати, 🔑Форма, 🔑Періодичність, Кількість_Днів, Коефіцієнт_Виплати 🔑Термін).

– атомарність: кожен атрибут повинен мати лише одне значення, а не множину значень.

В таблиці Громадянин атрибут Дані_паспорта має множину значень - прізвище ім’я по батькові, стать, дата народження, адреса проживання (область, район, місто, вулиця, дом, квартира), вид паспорта, серія та номер паспорта, а

атрибут Дані_свідоцтва_про_народження – прізвище ім’я по батькові, стать, дата народження, серія та номер свідоцтва, атрибут Дані_працевлаштування – область, район, місто, вулиця, дом, заробітня плата, атрибут Сімейний_дохід – сукупний дохід сім’ї, кількість членів сім’ї та дохід на особу. Розділивши дані атрибути на атомарні значення отримуємо відношення Громадянин (🔑Код_Громадянина, Прізвище, Ім’я, По_Батькові, Стать, Дата_народження, Адреса_проживання, Вид_паспорта, Номер_паспорта, Прізвище, Ім’я, По_Батькові, Дата_народження, Номер_свідоцтва, Телефон, Ідентифікаційний_код, Сукупний_дохід, Членів_домогосподарства, Дохід_на_особу) та бачимо, що дані повторюються, а це суперечить правилам першої нормальної форми. Видаливши дублюючі дані отримуємо відношення Громадянин (🔑Код_Громадянина, Прізвище, Ім’я, По_Батькові, Стать, Дата_народження, Адреса_проживання, Вид_паспорта, Номер_паспорта, Номер_свідоцтва, Телефон, Ідентифікаційний_код, Сукупний_дохід, Членів_домогосподарства, Дохід_на_особу).

Таблиця Журнал_Виплат має атрибут Дата, що включає рік, місяць, день, годину та хвилину виплати. Необхідно розділити даний атрибут на 2 стовпці: Дата (рік, місяць, день), Час (година, хвилина). Отримаємо відношення Журнал_Виплат (🔑Дата, Час, 🔑Код_Громадянина, 🔑Код_Пільги). При цьому атрибут Час не є ключовим стовпцем, оскільки неможлива ситуація, при якій один громадянин отримає декілька виплат з однієї пільги в один день.

В результаті проходження першої нормальної форми маємо відношення:

– Громадянин (🔑Код_Громадянина, Прізвище, Ім’я, По_Батькові, Стать, Дата_народження, Область, Район, Місто, Вулиця, Дом, Квартира, Вид_паспорта, Номер_паспорта, Номер_свідоцтва, Телефон, Ідентифікаційний_код, Сукупний_дохід, Членів_домогосподарства, Дохід_на_особу);

– Працевлаштування (🔑Код_Працевлаштування, Код_Громадянина, Посада, Область, Район, Місто, Вулиця, Дом, Заробітня_плата);

– Пільги (🔑Код_Пільги, Законодавство, Розмір, Код_Виплати);

– Пільги_Громадян (🔑Код_Громадянина, 🔑Код_Пільги);

– Виплата (Код_Виплати, 🔑Форма, 🔑Періодичність, Кількість_Днів, Коефіцієнт_Виплати 🔑Термін);

– Журнал_Виплат (🔑Дата, Час, 🔑Код_Громадянина, 🔑Код_Пільги).

Друга нормальна форма (2НФ, 2NF):

– схема бази даних повинна відповідати вимогам першої нормальної форми;

Схема бази даних знаходиться у першій нормальній формі.

– неключові атрибути повинні залежати від повного основного ключа (якщо він є складеним).

Атрибути Кількість_Днів та Коефіцієнт_Виплати таблиці Виплата залежать лише від атрибуту Періодичність, тобто не залежать від повного основного ключа. Додатково створимо унікальне числове поле Код_Періодичності для уникнення повторення даних у таблицях. Виконуємо декомпозицію та отримуємо два відношення: Виплата (Код_Виплати, 🔑Форма, 🔑Код_Періодичності, 🔑Термін) та Періодичність_Виплати (Код_Періодичності, 🔑Періодичність, Кількість_Днів, Коефіцієнт_Виплати).

В результаті проходження другої нормальної форми маємо відношення:

– Громадянин (🔑Код_Громадянина, Прізвище, Ім’я, По_Батькові, Стать, Дата_народження, Область, Район, Місто, Вулиця, Дом, Квартира, Вид_паспорта, Номер_паспорта, Номер_свідоцтва, Телефон, Ідентифікаційний_код, Сукупний_дохід, Членів_домогосподарства, Дохід_на_особу);

– Працевлаштування (🔑Код_Працевлаштування, Код_Громадянина, Посада, Область, Район, Місто, Вулиця, Дом, Заробітня_плата);

– Пільги (🔑Код_Пільги, Законодавство, Розмір, Код_Виплати);

– Пільги_Громадян (🔑Код_Громадянина, 🔑Код_Пільги);

– Виплата (Код_Виплати, 🔑Форма, 🔑Код_Періодичності, 🔑Термін);

– Періодичність_Виплати (Код_Періодичності, 🔑Періодичність, Кількість_Днів, Коефіцієнт_Виплати);

– Журнал_Виплат (🔑Дата, Час, 🔑Код_Громадянина, 🔑Код_Пільги).

Третя нормальна форма (3НФ, 3NF):

– схема бази даних повинна відповідати всім вимогам другої нормальної форми;

Схема бази даних знаходиться у другій нормальній формі.

– відсутність транзитивних залежностей неключових атрибутів від інших неключових атрибутів.

В таблиці Громадянин значення неключового атрибута Номер_свідоцтва залежить від неключового атрибута Дата_народження, оскільки, якщо громадянин досяг віку, з якого видається паспорт, то серія та номер свідоцтва не враховуються. Неключові атрибути Вид_паспорта, Номер_паспорта, Ідентифікаційний_код залежать від неключового атрибуту Дата_народження, оскільки, вид, серія, номер паспорта, ідентифікаційний код враховуються тільки при досягнені віку, з якого видається паспорт.

Для подолання транзитивних залежностей зробимо декомпозицію таблиці Громадянин на 3 відношення: Громадянин (🔑Код_Громадянина, Прізвище, Ім’я, По_Батькові, Стать, Дата_народження, Область, Район, Місто, Вулиця, Дом, Квартира, Телефон, Сукупний_дохід, Членів_домогосподарства, Дохід_на_особу), Громадянин_Паспорт (🔑Код_Громадянина, Вид_паспорта, Номер_паспорта, Ідентифікаційний_код), Громадянин_Свідоцтво (🔑Код_Громадянина, Номер_свідоцтва).

В результаті проходження третьої нормальної форми маємо відношення:

– Громадянин (🔑Код_Громадянина, Прізвище, Ім’я, По_Батькові, Стать, Дата_народження, Область, Район, Місто, Вулиця, Дом, Квартира, Телефон, Сукупний_дохід, Членів_домогосподарства, Дохід_на_особу);

– Громадянин_Паспорт (🔑Код_Громадянина, Вид_паспорта, Номер_паспорта, Ідентифікаційний_код);

– Громадянин_Свідоцтво (🔑Код_Громадянина, Номер_свідоцтва);

– Пільги (🔑Код_Пільги, Законодавство, Розмір, Код_Виплати);

– Пільги_Громадян (🔑Код_Громадянина, 🔑Код_Пільги);

– Виплата (Код_Виплати, 🔑Форма, 🔑Періодичність, 🔑Термін);

– Журнал_Виплат (🔑Дата, Час, 🔑Код_Громадянина, 🔑Код_Пільги);

– Періодичність_Виплати (Код_Періодичності, 🔑Періодичність, Кількість_Днів, Коефіцієнт_Виплати);

– Працевлаштування (🔑Код_Працевлаштування, Код_Громадянина, Посада, Область, Район, Місто, Вулиця, Дом, Заробітня_плата).

Нормальна форма Бойса-Кодда (НФБК):

– схема бази даних повинна відповідати всім вимогам третьої нормальної форми;

Схема бази даних знаходиться у третій нормальній формі.

– ключові атрибути складеного ключа не повинні залежати від неключових атрибутів.

Ключові атрибути складеного ключа таблиці Журнал_Виплат не залежать від неключового атрибута Час, а атрибут Код_Виплати таблиці Виплата не залежить від складеного ключа цієї таблиці, отже, модель знаходиться у нормальній формі Бойса-Кодда.

Четверта нормальна форма (4НФ, 4NF):

– схема бази даних повинна відповідати всім вимогам нормальної форми Бойса-Кодда;

Схема бази даних знаходиться у нормальній формі Бойса-Кодда.

– відсутність нетривіальних многозначних залежностей.

Відношення не мають нетривіальних многозначних залежностей. Отже, модель знаходиться у четвертій нормальній формі.

П’ята нормальна форма (5НФ, 5NF):

– схема бази даних повинна відповідати всім вимогам четвертої нормальної форми;

Схема бази даних знаходиться у четвертій нормальній формі.

– кожна нетривіальна залежність з’єднання (частина даних одного стовпця залежить від частини даних іншого стовпця) таблиці визначається потенційним ключом цієї таблиці.

Кожне нетривіальне з’єднання таблиць визначається потенційним ключем таблиці, отже, база даних знаходиться у 5НФ.

Нормальна форма домен/ключ:

– схема бази даних повинна відповідати всім вимогам п’ятої нормальної форми;

Схема бази даних знаходиться у п’ятій нормальній формі.

– дані в таблицях залежать лише від ключа та домену (набір можливих значень) атрибутів таблиць.

Визначимо домени атрибутів для відношення Громадянин:

– Код_Громадянина - обов’язкове унікальне числове поле;

– Прізвище - обов’язкове текстове поле, не може містити чисел або спецсимволи, окрім дефісу “-“ (можливі подвійні прізвища, наприклад, Квітка-Основ’яненко). Починається з великої літери;

– Ім’я - обов’язкове текстове поле, не може містити чисел або спецсимволи, окрім дефісу “-“ (можливі подвійні імена, наприклад, Софія-Саншейн). Починається з великої літери;

– По_Батькові - обов’язкове текстове поле, не може містити чисел та спецсимволів (подвійні імена по батькові неможливі згідно статті 147 Сімейного кодексу України: «Якщо ім’я батька є подвійним, то по батькові дитини може бути записано відповідно одного з цих імен»). Починається з великої літери;

– Стать - текстове поле, має лише 2 значення: Чоловік або Жінка;

– Дата_народження - обов’язкове поле, відповідність формату дати: чч.мм.рррр. Рік народження >= 1900; дата народження <= поточна дата;

– Область - обов’язкове текстове поле, не може містити чисел або спецсимволи, окрім дефісу “-“ (можливі подвійні назви областей);

– Район - обов’язкове текстове поле, не може містити чисел або спецсимволи, окрім дефісу “-“ (можливі подвійні назви районів);

– Місто - обов’язкове текстове поле, не може містити чисел або спецсимволи, окрім дефісу “-“ та прогалини “ “ (можливі подвійні назви міст);

– Вулиця - обов’язкове текстове поле, не може містити чисел або спецсимволи, окрім дефісу “-“ та прогалини “ “ (можливі подвійні назви вулиць);

– Дом - обов’язкове цілочисельне поле;

– Квартира - цілочисельне поле;

– Телефон - відповідність формату номеру мобільного телефону України: +3 (80X) XXX-XX-XX, де +380 - телефонний код країни, X - цифра від 0 до 9;

– Сукупний дохід - числове десяткове поле, не може бути менше нуля;

– Членів_домогосподарства - цілочисельне поле, не може бути менше 1;

– Дохід_на_особу - числове десяткове поле, не може бути менше нуля.

Визначимо домени атрибутів для відношення Працевлаштування:

– Код_Працевлаштування - обов’язкове унікальне числове поле;

– Код_Громадянина - обов’язкове унікальне числове поле, відповідає множині значень атрибута Код_Громадянина таблиці Громадянин;

– Посада - обов’язкове текстове поле;

– Область - обов’язкове текстове поле, не може містити чисел або спецсимволи, окрім дефісу “-“ (можливі подвійні назви областей);

– Район - обов’язкове текстове поле, не може містити чисел або спецсимволи, окрім дефісу “-“ (можливі подвійні назви районів);

– Місто - обов’язкове текстове поле, не може містити чисел або спецсимволи, окрім дефісу “-“ та прогалини “ “ (можливі подвійні назви міст);

– Вулиця - обов’язкове текстове поле, не може містити чисел або спецсимволи, окрім дефісу “-“ та прогалини “ “ (можливі подвійні назви вулиць);

– Дом - обов’язкове цілочисельне поле;

– Заробітня_плата - обов’язкове десяткове число.

Визначимо домени атрибутів для відношення Громадянин_Паспорт:

– Код_Громадянина - обов’язкове унікальне числове поле, відповідає множині значень атрибута Код_Громадянина таблиці Громадянин;

– Вид_паспорта - текстове поле без обмежень;

– Номер_паспорта - обов’язкове унікальне текстове поле;

– Ідентифікаційний_код - обов’язкове унікальне числове поле, що скаладається з десяти цифр.

Визначимо домени атрибутів для відношення Громадянин_Свідоцтво:

– Код_Громадянина - обов’язкове унікальне числове поле, відповідає множині значень атрибута Код_Громадянина таблиці Громадянин;

– Номер_свідоцтва - обов’язкове унікальне текстове поле.

Визначимо домени атрибутів для відношення Пільги:

– Код_Пільги - обов’язкове унікальне числове поле;

– Законодавство - обов’язкове унікальне текстове поле;

– Розмір - обов’язкове десяткове числове поле, не може бути менше нуля;

– Код_Виплати - обов’язкове числове поле, відповідає множині значень атрибута Код_Виплати таблиці Виплата.

Визначимо домени атрибутів для відношення Пільги_Громадян:

– Код_Громадянина - обов’язкове числове поле, відповідає множині значень атрибута Код_Громадянина таблиці Громадянин;

– Код_Пільги - обов’язкове числове поле, відповідає множині значень атрибута Код_Пільги таблиці Пільги.

Визначимо домени атрибутів для відношення Виплата:

– Код_Виплати - унікальне числове поле;

– Форма - обов’язкове текстове поле;

– Код_Періодичності - обов’язкове числове поле, відповідає множині значень атрибута Код_Періодичності таблиці Періодичність_Виплати;

– Термін - обов’язкове числове поле.

Визначимо домени атрибутів для відношення Періодичність_Виплати:

– Код_Періодичність_Термін - обов’язкове унікальне числове поле;

– Періодичність - обов’язкове текстове поле;

– Термін - необов’язкове числове поле.

Визначимо домени атрибутів для відношення Журнал_Виплат:

– Дата - обов’язкове поле, відповідність формату дати: чч.мм.рррр;

– Час - обов’язкове поле, відповідність формату гг:хх;

– Код_Громадянина - обов’язкове числове поле, відповідає множині значень атрибута Код_Громадянина таблиці Громадянин;

– Код_Пільги - обов’язкове числове поле, відповідає множині значень атрибута Код_Пільги таблиці Пільги.

Шоста нормальна форма (6НФ, 6NF):

– схема бази даних повинна відповідати всім вимогам нормальної форми домен/ключ;

Схема бази даних знаходиться у нормальній формі домен/ключ.

– таблиця повинна задовольняти усім нетривіальним залежностям з’єднання.

Таблиця Журнал_Виплат є хронологічної таблицею, отже, має сенс привести дану таблицю до 6НФ. Атрибути Дата та Час не є простими та можуть бути розбиті на менші одиниці інформації: Дата – день, місяць, рік; Час – година, хвилина. Однак, СУБД Microsoft Access підтримує тип даних «Дата й час» для роботи з датою та часом, внаслідок чого, відпадає необхідність декомпозиції даних атрибутів.

Отже, можна сказати, що таблиця Журнал_Виплат знаходиться у 6НФ, а приведення інших відношень до 6НФ призведе до втрати продуктивності бази даних.

Внаслідок нормалізації бази даних отримуємо такі відношення:

– Громадянин (🔑Код_Громадянина, Прізвище, Ім’я, По_Батькові, Стать, Дата_народження, Область, Район, Місто, Вулиця, Дом, Квартира, Телефон, Сукупний_дохід, Членів_домогосподарства, Дохід_на_особу);

– Громадянин_Паспорт (🔑Код_Громадянина, Вид_паспорта, Номер_паспорта, Ідентифікаційний_код);

– Громадянин_Свідоцтво (🔑Код_Громадянина, Номер_свідоцтва);

– Пільги (🔑Код_Пільги, Законодавство, Розмір, Код_Виплати);

– Пільги_Громадян (🔑Код_Громадянина, 🔑Код_Пільги);

– Виплата (Код_Виплати, 🔑Форма, 🔑Періодичність, 🔑Термін);

– Журнал_Виплат (🔑Дата, Час, 🔑Код_Громадянина, 🔑Код_Пільги);

– Періодичність_Виплати (Код_Періодичності, 🔑Періодичність, Кількість_Днів, Коефіцієнт_Виплати);

– Працевлаштування (🔑Код_Працевлаштування, Код_Громадянина, Посада, Область, Район, Місто, Вулиця, Дом, Заробітня_плата).

ДОДАТОК Б

Створення структури таблиць бази даних

Таблиця Громадянин:

За допомогою Конструктора таблиць створюємо таблицю Громадянин (Код_Громадянина, Прізвище, Ім’я, По_Батькові, Стать, Дата_народження, Телефон, Область, Район, Місто, Вулиця, Дом, Квартира, Сукупний_дохід, Членів_домогосподарства, Дохід_на_особу). В кожному рядку вписуємо один із атрибутів таблиці та вказуємо відповідний тип даних, що використовує даний атрибут.

Вказуємо ключове поле Код_Громадянина та встановлюємо тип даних Автонумерація. Таким чином, кожний громадянин, що додається до даної таблиці, автоматично отримує свій унікальний ідентифікатор, що дозволяє відрізнити кожний запис в таблиці. Поля таблиці Громадянин показані на рисунку Б.1.

Ім'я поля	Тип даних
Код_Громадянина	Автонумерація
Прізвище	Короткий текст
Ім'я	Короткий текст
По_Батькові	Короткий текст
Стать	Короткий текст
Дата_народження	Дата й час
Телефон	Короткий текст
Область	Короткий текст
Район	Короткий текст
Місто	Короткий текст
Вулиця	Короткий текст
Дом	Число
Квартира	Число
Сукупний_дохід	Число
Членів_домогосподарства	Число
Дохід_на_особу	Обчислюваний

Рисунок Б.1 – Поля таблиці Громадянин

Властивості полей таблиці Громадянин (пусті поля або поля без змін не вказані):

а) Код_Громадянина;

1) Розмір поля: Довге ціло число;

2) Нові значення: Поступово;

3) Індексовано: Так (Без повторень);

б) Прізвище, Ім’я, По_Батькові;

1) Розмір поля: 30;

2) Обов’язково: Так;

3) Дозволити нульову довжину: Ні;

4) Індексовано: Ні;

в) Стать

1) Розмір поля: 10;

2) Обов’язково: Так;

3) Дозволити нульову довжину: Ні;

г) Дата_народження;

1) Формат: Короткий формат дати;

2) Правило перевірки: >=#01.01.1900# And <=Date();

3) Обов’язково: Так;

4) Індексовано: Ні;

д) Телефон;

1) Розмір поля: 9;

2) Маска вводу: "+38(0"00") "000\-00\-00;;_;

3) Обов’язково: Ні;

4) Дозволити нульову довжину: Так;

5) Індексовано: Ні;

е) Область, Район, Місто, Вулиця;

1) Розмір поля: 30;

2) Обов’язково: Так;

3) Дозволити нульову довжину: Ні;

4) Індексовано: Ні;

ж) Дом;

1) Розмір поля: Ціле число;

2) Правило перевірки: >=1;

3) Текст перевірки: Номер дому не може бути менше 1.;

4) Обов’язково: Так;

5) Індексовано: Ні;

з) Квартира;

1) Розмір поля: Ціле число;

2) Правило перевірки: >=1;

3) Текст перевірки: Номер квартири не може бути менше 1.;

4) Обов’язково: Ні;

5) Індексовано: Ні;

и) Сукупний дохід;

1) Розмір поля: Подвійне значення;

2) Формат: # ##0.00" ₴";

3) Кількість знаків після коми: 2;

4) Значення за промовченням: 0;

5) Правило перевірки: >=0;

6) Текст перевірки: Рівень доходу не може бути менше нуля.;

к) Членів_домогосподарства;

1) Розмір поля: Ціле число;

2) Значення за промовченням: 1;

3) Правило перевірки: >=1;

4) Обов’язково: Так;

л) Дохід_на_особу;

1) Вираз: [Сукупний_дохід]/[Членів_домогосподарства];

2) Тип результату: Подвійне значення;

3) Формат: # ##0.00" ₴";

4) Кількість знаків після коми: 2.

На даному етапі визначені усі домени та обмеження для полів Код_Громадянина, Дата_народження, Телефон, Дом, Квартира, Сукупний_дохід, Членів_домогосподарства, Дохід_на_особу.

Для визначення доменів для поля Стать використаємо Майстер підстановок. У полі Тип даних атрибута Стать вибираємо Майстер підстановок… У вікні, що відкрилося обираємо «Я самостійно введу потрібні значення» та натискаємо Далі.

Вводимо необхідні дані та натискаємо Далі (рисунок Б.2).

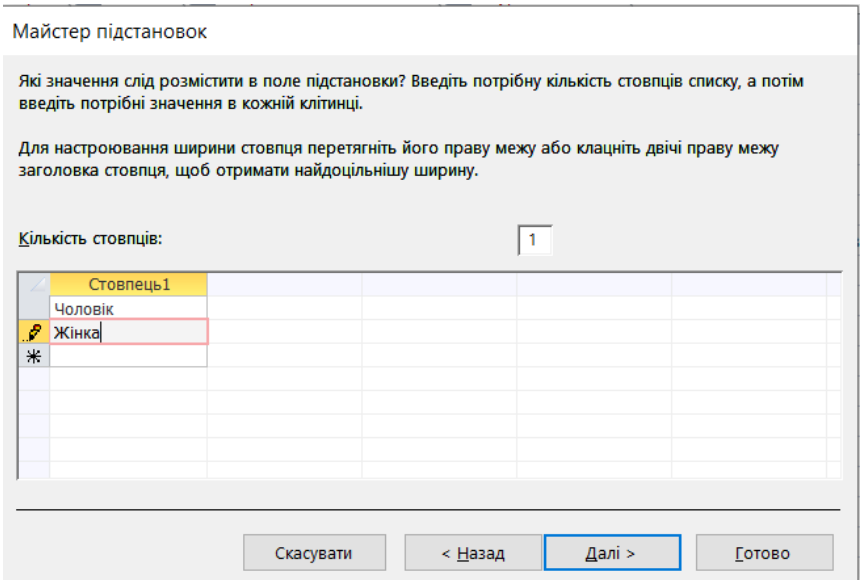


Рисунок Б.2 – Майстер підстановок для поля Стать

В наступному вікні ставимо галку навпроти Обмежити списком та натискаємо Готово.

Недовизначеними залишаються поля: Прізвище (не може починатися з прогалини " ", містити цифри, спецсимволи, окрім "-", починається з великої літери), Ім’я (не може починатися з прогалини " ", містити цифри, спецсимволи, окрім "-", починається з великої літери), По_Батькові (не може починатися з прогалини " ", містити цифри, спецсимволи, починається з великої літери), Область (не може починатися з прогалини " ", містити цифри, спецсимволи, окрім "-", починається з великої літери), Район (не може починатися з прогалини " ", містити цифри, спецсимволи, окрім "-", починається з великої літери), Місто (не може

починатися з прогалини " ", містити цифри, спецсимволи, окрім "-" та " ", починається з великої літери), Вулиця (не може починатися з прогалини " ", містити цифри, спецсимволи, окрім "-" та " ", починається з великої літери).

Для довизначення обмежень та доменів даних атрибутів недостатньо Конструктора таблиць. Дані обмеження можна ввести за допомогою регулярних виразів – рядок, що описує або збігається з множиною рядків, відповідно до набору спеціальних синтаксичних правил. Більшість високорівневих мов програмування підтримують регулярні вирази. СУБД Microsoft Access має вбудований редактор коду на мові Visual Basic for Applications (VBA), що дозволяє описати функції, які неможливо або досить складно описати за допомогою Конструктора таблиць або Конструктор макросів.

Відкрити редактор коду VBA можна за допомогою горячої клавіші Alt+F11 або в меню стрічки вибравши створення – Visual Basic.

Для підключення класів-обробників регулярних виразів у вікні редактора VBA в меню вибираємо Tools – References та ставимо галку навпроти Microsoft VBScript Regular Expressions 5.5 (рисунок Б.3).

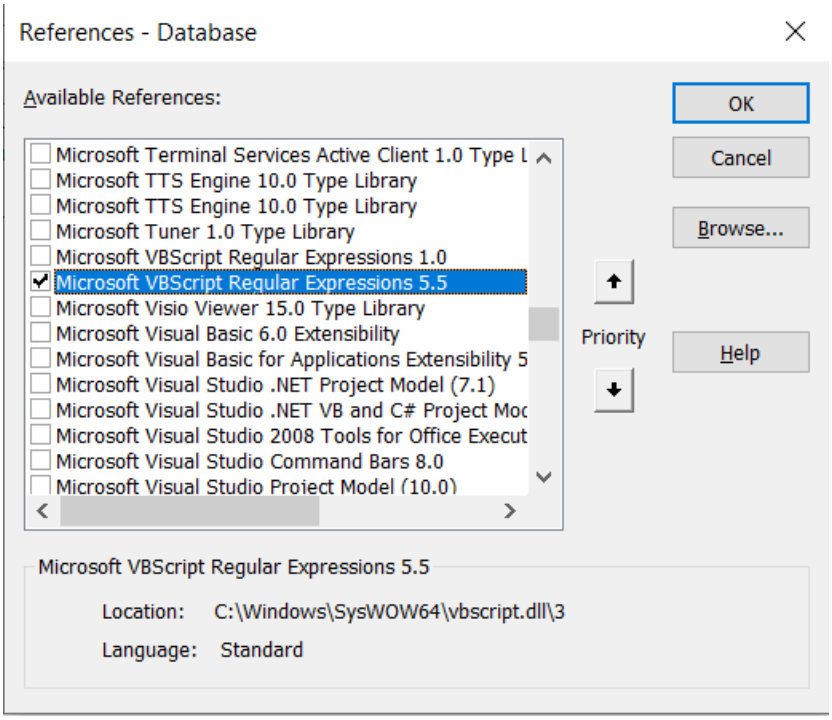


Рисунок Б.3 – Підключення регулярних виразів

Створимо два модулі:

– перевірка рядку – містить функції, що приймає в якості аргументу рядок та перевіряє його на відповідність деякому регулярному виразу та повертає значення булеве значення True, якщо рядок відповідає регулярному виразу, або False, якщо рядок не відповідає регулярному виразу;

Функція CheckOnlyUkranianSymbols приймає рядок і повертає True, якщо цей рядок містить лише українські літери, інакше повертає False.

Лістинг Б.1 – Функція перевірки рядку CheckOnlyUkranianSymbols

```
Public Function CheckOnlyUkranianSymbols(val As Variant) As Boolean
'Перевірка: рядок містить лише українські літери
    Dim myRegExp As New RegExp
    'Створюємо екземпляр RegExp
    myRegExp.Global = True
    'Якщо Global = True, то пошук ведеться по всьому рядку, якщо
    'False, то тільки до першого збігу
    myRegExp.Pattern                                                =
    "^[АаБбВвГгҐґДдЕеЄєЖжЗзИиІіЇїЙйКкЛлМмНнОоПпРрСсТтУуФфХхЦцЧчШшЩщ
    ЬьЮюЯя]+$"
    'Шаблон для пошуку
    CheckOnlyUkranianSymbols = myRegExp.Test(val)
End Function
```

Функція CheckOnlyUkranianSymbolsAndHyphen приймає рядок і повертає True, якщо цей рядок містить лише українські літери або знак дефісу “-“, інакше повертає False.

Лістинг Б.2 – Функція перевірки рядку CheckOnlyUkranianSymbolsAndHyphen

```
Public Function CheckOnlyUkranianSymbolsAndHyphen(val As Variant) As
Boolean
    'Перевірка: рядок містить лише українські літери або знак "-"
    Dim myRegExp As New RegExp
    'Створюємо екземпляр RegExp
    myRegExp.Global = True
    'Якщо Global = True, то пошук ведеться по всьому рядку, якщо
    'False, то тільки до першого збігу
    myRegExp.Pattern                                                =
    "(^[АаБбВвГгҐґДдЕеЄєЖжЗзИиІіЇїЙйКкЛлМмНнОоПпРрСсТтУуФфХхЦцЧчШшЩ
    щЬьЮюЯя]+$)|(^[АаБбВвГгҐґДдЕеЄєЖжЗзИиІіЇїЙйКкЛлМмНнОоПпРрСсТтУу
    ФфХхЦцЧчШшЩщЬьЮюЯя]+\-
    [АаБбВвГгҐґДдЕеЄєЖжЗзИиІіЇїЙйКкЛлМмНнОоПпРрСсТтУуФфХхЦцЧчШшЩщЬь
    ЮюЯя]+$)"
    'Шаблон для пошуку
```

```
    CheckOnlyUkranianSymbolsAndHyphen = myRegExp.Test(val)
End Function
```

Функція CheckOnlyUkranianSymbolsAndHyphenAndSpace приймає рядок і повертає True, якщо цей рядок містить лише українські літери або знак дефісу “-“ або знак прогалини “ “, інакше повертає False.

Лістинг Б.3 – Функція перевірки рядку CheckOnlyUkranianSymbolsAndHyphenAndSpace

```
Public Function CheckOnlyUkranianSymbolsAndHyphenAndSpace(val As
Variant) As Boolean
    'Перевірка: рядок містить лише українські літери або знак "-"
    Dim myRegExp As New RegExp
    'Створюємо екземпляр RegExp
    myRegExp.Global = True
    'Якщо Global = True, то пошук ведеться по всьому рядку, якщо
    'False, то тільки до першого збігу
    myRegExp.Pattern                                                =
    "(^[АаБбВвГгҐґДдЕеЄєЖжЗзИиІіЇїЙйКкЛлМмНнОоПпРрСсТтУуФфХхЦцЧчШшЩ
    щЬьЮюЯя]+$)|(^[АаБбВвГгҐґДдЕеЄєЖжЗзИиІіЇїЙйКкЛлМмНнОоПпРрСсТтУу
    ФфХхЦцЧчШшЩщЬьЮюЯя]+(\-|
    )[АаБбВвГгҐґДдЕеЄєЖжЗзИиІіЇїЙйКкЛлМмНнОоПпРрСсТтУуФфХхЦцЧчШшЩщЬ
    ьЮюЯя]+$)"
    'Шаблон для пошуку
    CheckOnlyUkranianSymbolsAndHyphenAndSpace = myRegExp.Test(val)
End Function
```

– нормалізація рядку – містить функції, що в якості аргументу приймають рядок та повертають новий форматований рядок.

Функція NormalString приймає рядок виду «василій петров» або «вАсилІй ПЕтРоВ» та форматує його до «Василій Петров».

Лістинг Б.4 – Функція нормалізації рядку

```
    'Код      даної      функції      запозичений      з      сайту
    http://msa.polarcom.ru/st/s0000260.htm
    Public Function NormalString(val As Variant) As Variant
    'Нормалізація рядку (ВАСЯ ПЕТРОВ або вася петров = Вася Петров)
    'Для виправлення введення Імен, Прізвищ (в тому числі і подвійних:
    морозов-пупкін = Морозов-Пупкін)
    '-------------------------------------------------------------
    On Error GoTo NormalString_Err
    Dim v As Variant
        If InStr(1, val, "-") > 0 Then
            v = Replace(val, "-", " - ")
            v = StrConv(v, 3)
            NormalString = Replace(v, " - ", "-")
```

```
    Else
        NormalString = StrConv(val, 3)
    End If

NormalString_End:
    On Error Resume Next
    Exit Function

NormalString_Err:
    NormalString = val 'Null
    'MsgBox "Error " & Err.Number & " (" & Err.Description & ")
in procedure NormalString, line " & Erl & ".", vbCritical
    Err.Clear
    Resume NormalString_End
End Function
```

Для використання даних функцій необхідно створити макрос, що буде викликатися кожний раз при зміні даних у таблиці Громадянин. Для цього у відкритому Конструкторі таблиці Громадянин на стрічці вибираємо Конструктор – Створити макроси даних – Перед зміненням.

За допомогою Конструктора макросів створюємо макрос, що спочатку перевіряє поля Прізвище, Ім’я, По_Батькові, Область, Район, Місто, Вулиця визиваючи відповідну функцію у модулі Перевірка рядку для перевірки рядку символів на відповідність регулярному виразу, інакше виводить повідомлення про помилку. Зразок макросу для поля Прізвище показаний на рисунку Б.4.

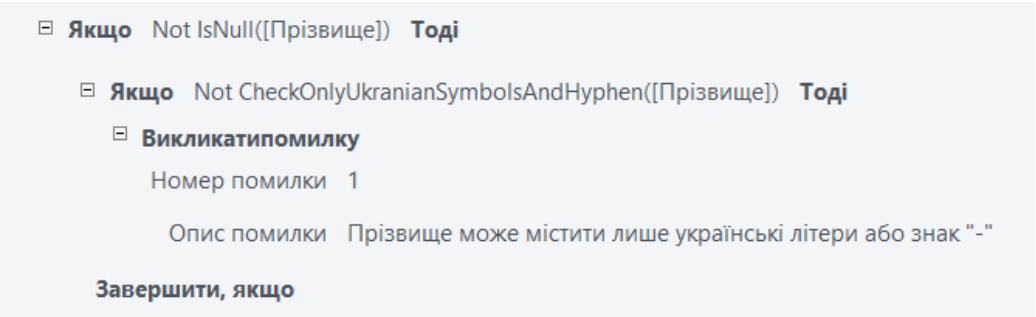


Рисунок Б.4 – Перевірка поля Прізвище

Після перевірки кожне поле форматується за допомогою двох функцій – функції NormalString модуля Нормалізація рядку та вбудованої функції Trim, що

приймає в якості аргументу рядок та видаляє прогалини на початку та кінці рядка, якщо вони є. Наприклад, Trim(“ Олег Шатов “) поверне рядок “Олег Шатов”. Зразок нормалізації для поля Прізвище показаний на рисунку Б.5.

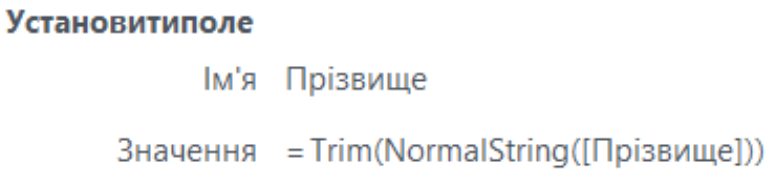


Рисунок Б.5 – Форматування поля Прізвище

Повний лістинг макросу представлено у вигляді XML-опису в додатку В.

Таблиця Громадянин_Паспорт:

За допомогою Конструктора таблиць створюємо відношення Громадянин_Паспорт (Код_Громадянина, Вид_паспорта, Номер_паспорта, Ідентифікаційний_код). Поля таблиці Громадянин_Паспорт показані на рисунку Б.6

Ім'я поля	Тип даних
Код_Громадянина	Число
Вид_паспорта	Короткий текст
Номер_паспорта	Короткий текст
Ідентифікаційний_код	Короткий текст

Рисунок Б.6 – Поля таблиці Громадянин_Паспорт

Властивості полей таблиці Громадянин_Паспорт (пусті поля або поля без змін не вказані):

а) Код_Громадянина;

1) Розмір поля: Довге ціле число;

2) Обов’язково: Так;

3) Індексовано: Так (Без повторень);

б) Вид_паспорта;

1) Розмір поля: 30;

2) Обов’язково: Так;

3) Дозволити нульову довжину: Ні;

4) Індексовано: Так (Повторення дозволені);

в) Номер_паспорта;

1) Розмір поля: 14;

2) Обов’язково: Так;

3) Дозволити нульову довжину: Ні;

4) Індексовано: Так (Без повторень);

г) Ідентифікаційний_код;

1) Розмір поля: 10;

2) Маска вводу: 0000000000;;_;

3) Обов’язково: Так;

4) Дозволити нульову довжину: Ні;

5) Індексовано: Так (Без повторень).

Для полів Код_Громадянина, Ідентифікаційний_код визначено усі обмеження.

Для визначення доменів для поля Вид_паспорта використаємо Майстер підстановок (рисунок Б.7).

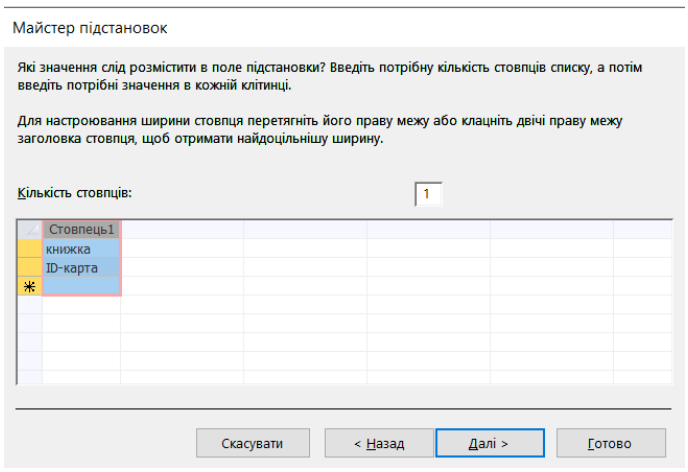


Рисунок Б.7 – Майстер підстановок для поля Вид_паспорта

Формат поля Номер_паспорта залежить від поля Вид_паспорта. Тому для перевірки правильності вводу номеру паспорта створимо дві функції в модулі Перевірка рядку: функцію CheckPassportBook для перевірки номеру паспорта-книжки та функцію CheckPassportIDCard для перевірки номеру паспорта-ID-карти.

Лістинг Б.6 – Функція перевірки номера паспорта-книжки

```
Public Function CheckPassportBook(val As Variant) As Boolean
    'Перевірка рядку на відповідність формату АБ123456
    Dim myRegExp As New RegExp
    'Створюємо екземпляр RegExp
    myRegExp.Global = True
    'Якщо Global = True, то пошук ведеться по всьому рядку, якщо
    'False, то тільки до першого збігу
    myRegExp.Pattern                                                =
    "^[АаБбВвГгҐґДдЕеЄєЖжЗзИиІіЇїЙйКкЛлМмНнОоПпРрСсТтУуФфХхЦцЧчШшЩщ
    ЬьЮюЯя]{2}[0-9]{6}$"
    CheckPassportBook = myRegExp.Test(val)
End Function
```

Лістинг Б.7 – Функція перевірки номера паспорта-ID-карти

```
Public Function CheckPassportIDCard(val As Variant) As Boolean
    'Перевірка рядку на відповідність формату 12345678-12345 або
    '1234567812345 або 12345678 12345
    Dim myRegExp As New RegExp
    'Створюємо екземпляр RegExp
    myRegExp.Global = True
    'Якщо Global = True, то пошук ведеться по всьому рядку, якщо
    'False, то тільки до першого збігу
    myRegExp.Pattern = "^[0-9]{8}(\-| )?[0-9]{5}$"
    CheckPassportIDCard = myRegExp.Test(val)
End Function
```

Створюємо макрос:перед зміненням для таблиці Громадянин_Паспорт для виклику описаних функцій перед кожним зміненням даних у таблиці. В залежності від значення поля Вид_паспорта викликається потрібна функція: CheckPassportBook або CheckPassportIDCard.

Після вдалої перевірки номер паспорту нормалізується до виду «АБ123456», якщо Вид_паспорта «книжка» або нормалізується до виду «12345678-12345», якщо Вид_паспорта «ID-карта».

Для нормалізації паспорта-книжки викликається функція UCase, яка переводить усі символи у верхній регістр. Для нормалізації паспорта-ID-карти

створена функція NormalPassportIDCard у модулі Нормалізація рядку. Дана функція нормалізує рядок до виду «12345678-12345».

Лістинг Б.8 – Функція нормалізації паспорта-ID-карти

```
Public Function NormalPassportIDCard(val As Variant) As Variant
    'Нормалізація номера паспорту ID-карти (1234567812345 або
    '12345678 12345 = 12345678-12345)
    NormalPassportIDCard = Left(val, 8) & "-" & Right(val, 5)
End Function
```

Повний лістинг макросу представлено у вигляді XML-опису в додатку Г.

Таблиця Громадянин_Свідоцтво:

За допомогою Конструктора таблиць створюємо відношення Громадянин_Свідоцтво (Код_Громадянина, Номер_свідоцтва). Вказуємо ключове поле Код_Громадянина. Поля таблиці Громадянин_Свідоцтво показані на рисунку Б.8.

	Ім'я поля	Тип даних
🔑	Код_Громадянина	Число
	Номер_свідоцтва	Короткий текст

Рисунок Б.8 – Поля таблиці Громадянин_Свідоцтво

Властивості полей таблиці (пусті поля або поля без змін не вказані):

а) Код_Громадянина;

1) Розмір поля: Довге ціле число;

2) Обов’язково: Так;

3) Індексовано: Так (Без повторень);

б) Номер_свідоцтва;

1) Розмір поля: 9;

2) Маска вводу: "Серія ">L"-"LL<" №"000000;;_;

3) Обов’язково: Так;

4) Дозволити нульову довжину: Ні;

5) Індексовано: Так (Без повторень).

Таблиця Пільги:

За допомогою Конструктора таблиць створюємо відношення Пільги (Код_Пільги, Законодавство, Розмір, Код_Виплати). Поля таблиці Пільги показані на рисунку Б.9).

	Ім'я поля	Тип даних
🔑	Код_Пільги	Автонумерація
	Законодавство	Короткий текст
	Розмір	Число
	Код_Виплати	Число

Рисунок Б.9 – Поля таблиці Пільги

Властивості полей таблиці (пусті поля або поля без змін не вказані):

а) Код_Пільги;

1) Розмір поля: Довге ціле число;

2) Нові значення: Поступово;

3) Індексовано: Так (Без повторень);

б) Законодавство;

1) Розмір поля: 50;

2) Обов’язково: Так;

3) Дозволити нульову довжину: Ні;

4) Індексовано: Так (Без повторень);

в) Розмір;

1) Розмір поля: Подвійне значення;

2) Формат: 0" %";

3) Кількість знаків після коми: 2;

4) Значення за промовченням: 0;

5) Правило перевірки: >=0;

6) Текст перевірки: Значення розміру не може бути менше нуля.;

7) Обов’язково: Так;

8) Індексовано: Ні;

г) Код_Виплати;

1) Розмір поля: Довге ціле число;

2) Обов’язково: Так;

3) Індексовано: Так (Повторення дозволені).

Таблиця Пільги_Громадян:

За допомогою Конструктора таблиць створюємо відношення Пільги_Громадян (🔑Код_Громадянина, 🔑Код_Пільги). Поля таблиці Пільги_Громадян показані на рисунку Б.10.

	Ім'я поля	Тип даних
🔑	Код_Громадянина	Число
🔑	Код_Пільги	Число

Рисунок Б.10 – Поля таблиці Пільги_Громадян

Властивості полей таблиці (пусті поля або поля без змін не вказані):

а) Код_Громадянина, Код_Пільги;

1) Розмір поля: Довге ціле число;

2) Обов’язково: Так;

3) Індексовано: Так (Повторення дозволені).

Таблиця Виплата:

За допомогою Конструктора таблиць створюємо відношення Виплата (Код_Виплати, 🔑Форма, 🔑Код_Періодичності, 🔑Термін). Поля таблиці Виплата показані на рисунку Б.11.

	Ім'я поля	Тип даних
	Код_Виплати	Автонумерація
🔑	Форма	Короткий текст
🔑	Код_Періодичності	Число
🔑	Термін	Число

Рисунок Б.11 – Поля таблиці Виплата

Властивості полей таблиці (пусті поля або поля без змін не вказані):

а) Код_Виплати;

1) Розмір поля: Довге ціле число;

2) Нові значення: Поступово;

3) Індексовано: Так (Без повторень);

б) Форма;

1) Розмір поля: 30;

2) Обов’язково: Так;

3) Дозволити нульову довжину: Ні;

4) Індексовано: Так (Повторення дозволені);

в) Код_Періодичності;

1) Розмір поля: Довге ціле число;

2) Обов’язково: Так;

3) Індексовано: Ні;

г) Термін;

1) Розмір поля: Ціле число;

2) Формат: 0" дн.";

3) Правило перевірки: >=0;

4) Текст перевірки: Значення терміну повинне бути більше нуля.;

5) Обов’язково: Так.

В якості типу даних для поля Форма вибираємо Майстер_Підстановок…, обмежуємо дане поле списком вводу (рисунок Б.12).

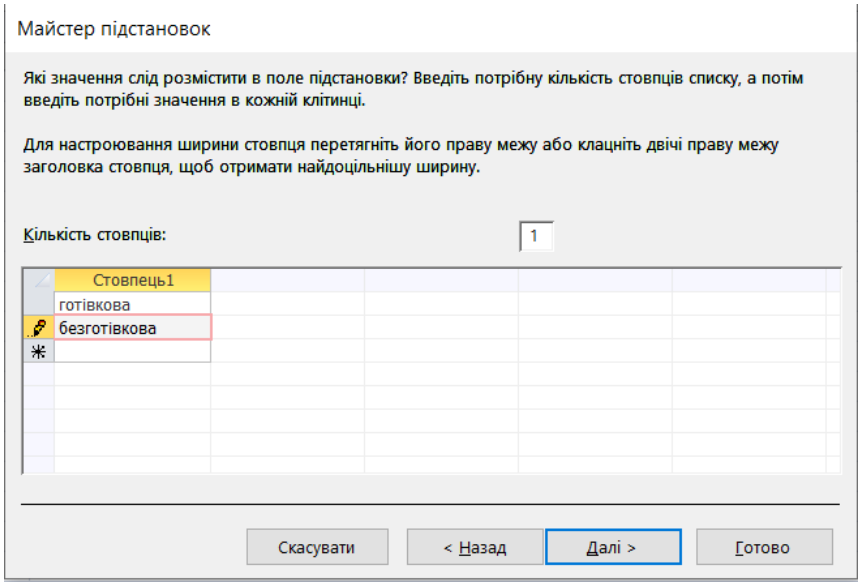


Рисунок Б.12 – Майстер підстановок для поля Форма

Таблиця Періодичність_Виплати:

За допомогою Конструктора таблиць створюємо відношення Періодичність_Виплати (Код_Періодичності, Періодичність, Кількість_днів, Коефіцієнт_Виплати). Поля таблиці Періодичність_Виплати показані на рисунку Б.13.

Ім'я поля	Тип даних
Код_Періодичності	Автонумерація
Періодичність	Короткий текст
Кількість_днів	Число
Коефіцієнт_Виплати	Обчислюваний

Рисунок Б.13 – Поля таблиці Періодичність_Виплати

Властивості полей таблиці (пусті поля або поля без змін не вказані):

а) Код_Періодичності;

1) Розмір поля: Довге ціле число;

2) Нові значення: Поступово;

3) Індексовано: Так (Без повторень);

б) Періодичність;

1) Розмір поля: 30;

2) Обов’язково: Так;

3) Дозволити нульову довжину: Ні;

4) Індексовано: Так (Без повторень);

в) Кількість днів;

1) Розмір поля: Ціле число;

2) Обов’язково: Ні;

г) Коефіцієнт_Виплати;

1) Вираз: IIf([Періодичність]="одноразово";1;1/(365/[Кількість_днів]));

2) Тип результату: Подвійне значення;

3) Формат: Фіксований;

4) Кількість знаків після коми: 4.

Таблиця Журнал_Виплат:

За допомогою Конструктора таблиць створюємо відношення Журнал_Виплат (🔑Дата, Час, 🔑Код_Громадянина, 🔑Код_Пільги). Поля таблиці Журнал_Виплат показані на рисунку Б.14.

	Ім'я поля	Тип даних
🔑	Дата	Дата й час
	Час	Дата й час
🔑	Код_Громадянина	Число
🔑	Код_Пільги	Число

Рисунок Б.14 – Поля таблиці Журнал_Виплат

Властивості полей таблиці (пусті поля або поля без змін не вказані):

а) Дата;

1) Формат: Короткий формат дати;

2) Значення за промовченням: =Date();

3) Правило перевірки: <=Now();

4) Текст перевірки: Значення дати не може бути більше поточної дати;

5) Обов’язково: Так;

б) Час;

1) Формат: Довгий формат часу;

2) Значення за промовченням: =Time();

3) Обов’язково: Так;

в) Код_Громадянина, Код_Пільги;

1) Розмір поля: Довге ціле число;

2) Обов’язково: Так.

Для перевірки введення значення дати та часу не вище поточної створимо макрос: перед зміненням, який перевіряє правильність вводу дати та часу за допомогою формули:

$$([\text{Дата}] + [\text{Час}]) > Now().$$

Таблиця Працевлаштування:

За допомогою Конструктора таблиць створюємо відношення Працевлаштування (Код_Працевлаштування, Код_Громадянина, Посада, Область, Район, Місто, Вулиця, Дом, Заробітня_плата). Поля таблиці Працевлаштування показані на рисунку Б.15.

	Ім'я поля	Тип даних
🔑	Код_Працевлаштування	Автонумерація
	Код_Громадянина	Число
	Посада	Короткий текст
	Область	Короткий текст
	Район	Короткий текст
	Місто	Короткий текст
	Вулиця	Короткий текст
	Дом	Число
	Заробітня_плата	Число

Рисунок Б.15 – Поля таблиці Працевлаштування

Властивості полей таблиці (пусті поля або поля без змін не вказані):

а) Код_Працевлаштування;

1) Розмір поля: Довге ціле число;

2) Нові значення: Поступово;

3) Так (Без повторень);

б) Код_Громадянина;

1) Розмір поля: Довге ціле число;

2) Обов’язково: Так;

3) Індексовано: Так (Повторення дозволені);

в) Посада;

1) Розмір поля: 30;

2) Обов’зяково: Так;

3) Дозволити нульову довжину: Ні;

г) Область, Район, Місто, Вулиця;

1) Розмір поля: 30;

2) Обов’язково: Ні;

д) Дом;

1) Розмір поля: Ціле число;

2) Правило перевірки: >=1;

3) Текст перевірки: Номер дому не може бути менше 1.;

4) Обов’язково: Ні;

е) Заробітня плата;

1) Розмір поля: Подвійне значення;

2) Формат: # ##0.00" ₴";

3) Значення за промовченням: 0;

4) Кількість знаків після коми: 2;

5) Обов’язково: Так;

Створимо макрос: перед зміненням для перевірки та редагування полів Область, Район, Місто, Вулиця за допомогою раніше визначених функцій VBA. Лістинг макросу у вигляді XML-опису подано в додатку Д.

Після створення усіх таблиць визначаємо домени для полів таблиць, що зв’язані з полями іншими таблиць за допомогою підстановки, в якості джерела рядків використовуючи SQL-запити та обмежуючи дане поле списком підстановки.

Громадянин_Паспорт.Код_Громадянина - SELECT Код_Громадянина AS Код, Прізвище, [Ім'я], По_Батькові, Дата_Народження FROM Громадянин ORDER BY Прізвище, [Ім'я], По_Батькові, Дата_Народження.

Громадянин_Свідоцтво.Код_Громадянина - SELECT Код_Громадянина AS Код, Прізвище, [Ім'я], По_Батькові, Дата_Народження FROM Громадянин ORDER BY Прізвище, [Ім'я], По_Батькові, Дата_Народження.

Журнал_Виплат.Код_Громадянина - SELECT Код_Громадянина AS Код, Прізвище, [Ім'я], По_Батькові, Дата_Народження FROM Громадянин ORDER BY Прізвище, [Ім'я], По_Батькові, Дата_Народження.

Журнал_Виплат.Код_Пільги - SELECT Код_Пільги AS Код, Законодавство FROM Пільги ORDER BY Законодавство.

Виплата.Код_Періодичності - SELECT Код_Періодичності AS Код, Періодичність FROM Періодичність_Виплати.

Пільги.Код_Виплати - SELECT Код_Виплати AS Код, Форма, (SELECT Періодичність FROM Періодичність_Виплати WHERE Код_Періодичності = Виплата.Код_Періодичності) AS Періодичність, Термін FROM Виплата.

Пільги_Громадян.Код_Громадянина - SELECT Код_Громадянина AS Код, Прізвище, [Ім'я], По_Батькові, Дата_Народження FROM Громадянин ORDER BY Прізвище, [Ім'я], По_Батькові, Дата_Народження.

Пільги_Громадян.Код_Пільги - SELECT Код_Пільги AS Код, Законодавство FROM Пільги ORDER BY Законодавство.

Працевлаштування.Код_Громадянина - SELECT Код_Громадянина AS Код, Прізвище, [Ім'я], По_Батькові, Дата_Народження FROM Громадянин ORDER BY Прізвище, [Ім'я], По_Батькові, Дата_Народження.

ДОДАТОК В

XML-опис макросу до зміни даних таблиці Громадянин

Лістинг В.1 – XML-опис макросу до зміни даних таблиці Громадянин

```
<?xml version="1.0" encoding="UTF-16" standalone="no"?>
<DataMacros
xmlns="http://schemas.microsoft.com/office/accessservices/2009/11/application">
 <DataMacro Event="BeforeChange">
  <Statements>
   <StatementGroup Description="Перевірка на вміст цифр/спец.символів">
    <Statements>
     <ConditionalBlock>
      <If>
       <Condition>Not IsNull([Прізвище])</Condition>
       <Statements>
        <ConditionalBlock>
         <If>
          <Condition>Not CheckOnlyUkranianSymbolsAndHyphen([Прізвище])</Condition>
          <Statements>
           <Action Name="RaiseError">
            <Argument Name="Number">1</Argument>
            <Argument Name="Description">Прізвище може містити лише українські літери або знак "-"</Argument>
           </Action>
          </Statements>
         </If>
        </ConditionalBlock>
       </Statements>
      </If>
     </ConditionalBlock>
     <ConditionalBlock>
      <If>
       <Condition>Not IsNull([Ім'я])</Condition>
       <Statements>
        <ConditionalBlock>
         <If>
          <Condition>Not CheckOnlyUkranianSymbolsAndHyphen([Ім'я])</Condition>
          <Statements>
           <Action Name="RaiseError">
            <Argument Name="Number">2</Argument>
            <Argument Name="Description">Ім'я може містити лише українські літери або знак "-"</Argument>
           </Action>
          </Statements>
```

```
          </If>
         </ConditionalBlock>
        </Statements>
       </If>
      </ConditionalBlock>
      <ConditionalBlock>
       <If>
        <Condition>Not IsNull([По_Батькові])</Condition>
        <Statements>
         <ConditionalBlock>
          <If>
           <Condition>Not
CheckOnlyUkranianSymbols([По_Батькові])</Condition>
           <Statements>
            <Action Name="RaiseError">
             <Argument Name="Number">3</Argument>
             <Argument   Name="Description">Ім'я   по   батькові   може
містити лише українські літери</Argument>
            </Action>
           </Statements>
          </If>
         </ConditionalBlock>
        </Statements>
       </If>
      </ConditionalBlock>
      <ConditionalBlock>
       <If>
        <Condition>Not IsNull([Область])</Condition>
        <Statements>
         <ConditionalBlock>
          <If>
           <Condition>Not
CheckOnlyUkranianSymbolsAndHyphen([Область])</Condition>
           <Statements>
            <Action Name="RaiseError">
             <Argument Name="Number">4</Argument>
             <Argument  Name="Description">Область  може  містити  лише
українські літери або знак "-"</Argument>
            </Action>
           </Statements>
          </If>
         </ConditionalBlock>
        </Statements>
       </If>
      </ConditionalBlock>
      <ConditionalBlock>
       <If>
        <Condition>Not IsNull([Район])</Condition>
        <Statements>
         <ConditionalBlock>
          <If>
           <Condition>Not
CheckOnlyUkranianSymbolsAndHyphen([Район])</Condition>
```

```
          <Statements>
           <Action Name="RaiseError">
            <Argument Name="Number">5</Argument>
            <Argument  Name="Description">Район  може  містити  лише
українські літери або знак "-"</Argument>
           </Action>
          </Statements>
         </If>
        </ConditionalBlock>
       </Statements>
      </If>
     </ConditionalBlock>
     <ConditionalBlock>
      <If>
       <Condition>Not IsNull([Місто])</Condition>
       <Statements>
        <ConditionalBlock>
         <If>
          <Condition>Not
CheckOnlyUkranianSymbolsAndHyphenAndSpace([Місто])</Condition>
          <Statements>
           <Action Name="RaiseError">
            <Argument Name="Number">6</Argument>
            <Argument  Name="Description">Місто  може  містити  лише
українські літери або знак "-" або прогалину ""</Argument>
           </Action>
          </Statements>
         </If>
        </ConditionalBlock>
       </Statements>
      </If>
     </ConditionalBlock>
     <ConditionalBlock>
      <If>
       <Condition>Not IsNull([Вулиця])</Condition>
       <Statements>
        <ConditionalBlock>
         <If>
          <Condition>Not
CheckOnlyUkranianSymbolsAndHyphenAndSpace([Вулиця])</Condition>
          <Statements>
           <Action Name="RaiseError">
            <Argument Name="Number">7</Argument>
            <Argument  Name="Description">Вулиця  може  містити  лише
українські літери або знак "-"</Argument>
           </Action>
          </Statements>
         </If>
        </ConditionalBlock>
       </Statements>
      </If>
     </ConditionalBlock>
    </Statements>
```

```
   </StatementGroup>
   <StatementGroup Description="Форматування рядку">
    <Statements>
     <Action Name="SetField">
      <Argument Name="Field">Прізвище</Argument>
      <Argument
Name="Value">Trim(NormalString([Прізвище]))</Argument>
     </Action>
     <Action Name="SetField">
      <Argument Name="Field">Ім'я</Argument>
      <Argument Name="Value">Trim(NormalString([Ім'я]))</Argument>
     </Action>
     <Action Name="SetField">
      <Argument Name="Field">По_Батькові</Argument>
      <Argument
Name="Value">Trim(NormalString([По_Батькові]))</Argument>
     </Action>
     <Action Name="SetField">
      <Argument Name="Field">Область</Argument>
      <Argument
Name="Value">Trim(NormalString([Область]))</Argument>
     </Action>
     <Action Name="SetField">
      <Argument Name="Field">Район</Argument>
      <Argument Name="Value">Trim(NormalString([Район]))</Argument>
     </Action>
     <Action Name="SetField">
      <Argument Name="Field">Місто</Argument>
      <Argument Name="Value">Trim(NormalString([Місто]))</Argument>
     </Action>
     <Action Name="SetField">
      <Argument Name="Field">Вулиця</Argument>
      <Argument Name="Value">Trim(NormalString([Вулиця]))</Argument>
     </Action>
    </Statements>
   </StatementGroup>
  </Statements>
 </DataMacro>
</DataMacros>
```

ДОДАТОК Г

XML-опис макросу до зміни даних таблиці Громадянин_Паспорт

Лістинг Г.1 – XML-опис макросу до зміни даних в таблиці Громадянин_Паспорт

```
<?xml version="1.0" encoding="UTF-16" standalone="no"?>
<DataMacros
xmlns="http://schemas.microsoft.com/office/accessservices/2009/11/application">
 <DataMacro Event="BeforeChange">
  <Statements>
   <StatementGroup Description="Перевірка номеру паспорта">
    <Statements>
     <ConditionalBlock>
      <If>
       <Condition>Not IsNull([Номер_паспорта])</Condition>
       <Statements>
        <ConditionalBlock>
         <If>
          <Condition>[Вид_паспорта]="книжка"</Condition>
          <Statements>
           <ConditionalBlock>
            <If>
             <Condition>Not
CheckPassportBook([Номер_паспорта])</Condition>
             <Statements>
              <Action Name="RaiseError">
               <Argument Name="Number">11</Argument>
               <Argument  Name="Description">Не  відповідає  формату
номера паспорта "книжка"</Argument>
              </Action>
             </Statements>
            </If>
           </ConditionalBlock>
          </Statements>
         </If>
         <ElseIf>
          <Condition>[Вид_паспорта]="ID-карта"</Condition>
          <Statements>
           <ConditionalBlock>
            <If>
             <Condition>Not
CheckPassportIDCard([Номер_паспорта])</Condition>
             <Statements>
              <Action Name="RaiseError">
               <Argument Name="Number">12</Argument>
```

```
                <Argument  Name="Description">Не  відповідає  формату
номера паспорта "ID-карта"</Argument>
               </Action>
              </Statements>
             </If>
            </ConditionalBlock>
           </Statements>
          </ElseIf>
         </ConditionalBlock>
        </Statements>
       </If>
      </ConditionalBlock>
     </Statements>
    </StatementGroup>
    <StatementGroup Description="Нормалізація номеру паспорта">
     <Statements/>
    </StatementGroup>
    <ConditionalBlock>
     <If>
      <Condition>[Вид_паспорта]="книжка"</Condition>
      <Statements>
       <Action Name="SetField">
        <Argument Name="Field">Номер_паспорта</Argument>
        <Argument Name="Value">UCase([Номер_паспорта])</Argument>
       </Action>
      </Statements>
     </If>
     <ElseIf>
      <Condition>[Вид_паспорта]="ID-карта"</Condition>
      <Statements>
       <Action Name="SetField">
        <Argument Name="Field">Номер_паспорта</Argument>
        <Argument
Name="Value">NormalPassportIDCard([Номер_паспорта])</Argument>
       </Action>
      </Statements>
     </ElseIf>
    </ConditionalBlock>
   </Statements>
  </DataMacro>
</DataMacros>
```

ДОДАТОК Д

XML-опис макросу до зміни даних таблиці Журнал_Виплат

Лістинг Д.1 – XML-опис макросу до зміни даних таблиці Журнал_Виплат

```
<?xml version="1.0" encoding="UTF-16" standalone="no"?>
<DataMacros
xmlns="http://schemas.microsoft.com/office/accessservices/2009/11/ap
plication">
 <DataMacro Event="BeforeChange">
  <Statements>
   <ConditionalBlock>
    <If>
     <Condition>Not IsNull([Дата])</Condition>
     <Statements>
      <ConditionalBlock>
       <If>
        <Condition>Not IsNull([Час])</Condition>
        <Statements>
         <ConditionalBlock>
          <If>
           <Condition>([Дата]+[Час])&gt;Now()</Condition>
           <Statements>
            <Action Name="RaiseError">
             <Argument Name="Number">41</Argument>
             <Argument Name="Description">Дата та час не можуть бути
більше поточного.</Argument>
            </Action>
           </Statements>
          </If>
         </ConditionalBlock>
        </Statements>
       </If>
      </ConditionalBlock>
     </Statements>
    </If>
   </ConditionalBlock>
  </Statements>
 </DataMacro>
</DataMacros>
```

ДОДАТОК Е

XML-опис макросу до зміни даних таблиці Працевлаштування

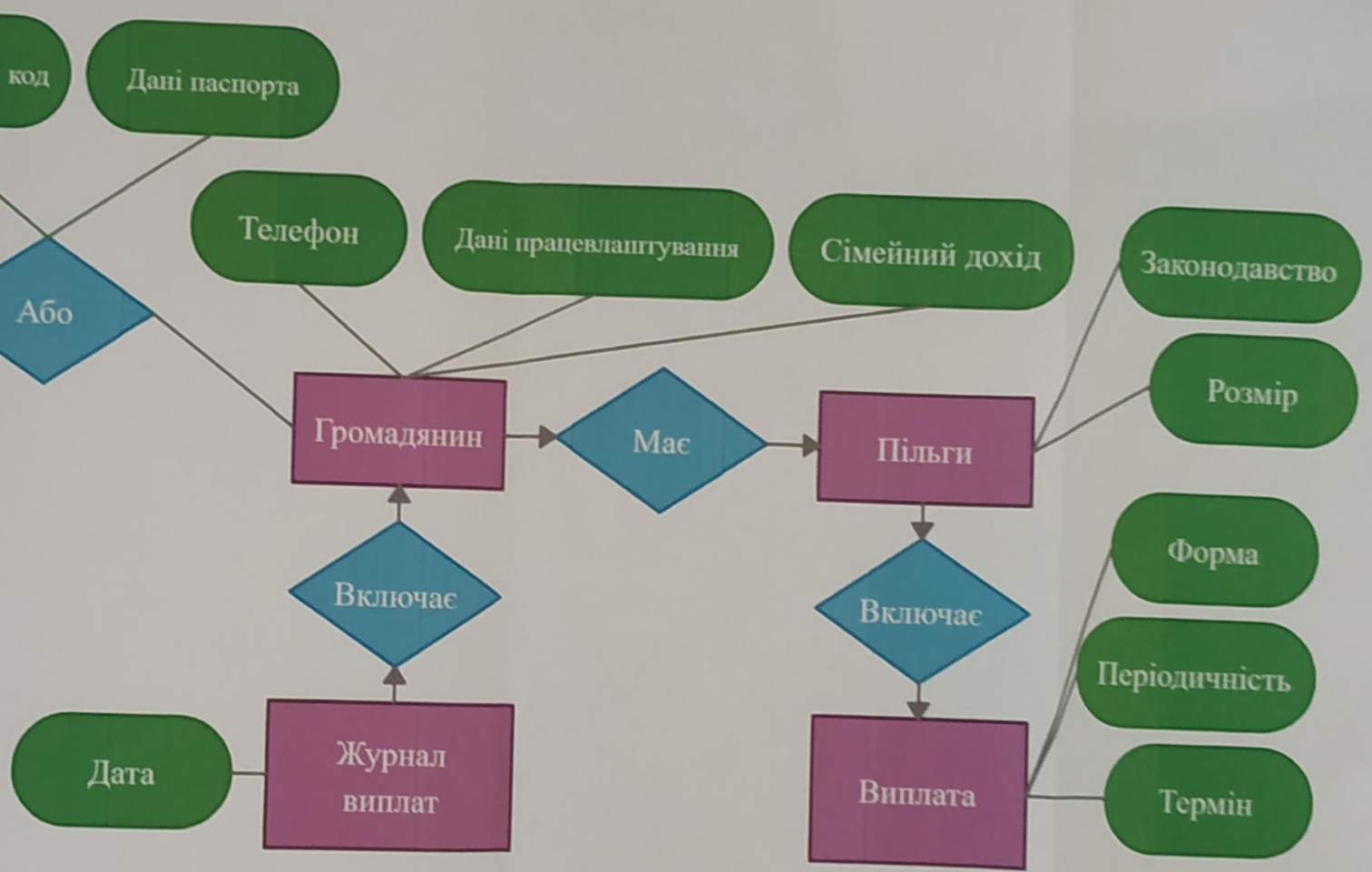
Лістинг Е.1 – XML-опис макросу до зміни даних таблиці Працевлаштування

```
<?xml version="1.0" encoding="UTF-16" standalone="no"?>
<DataMacros
xmlns="http://schemas.microsoft.com/office/accessservices/2009/11/application">
 <DataMacro Event="BeforeChange">
  <Statements>
   <StatementGroup Description="Перевірка на вміст цифр/спец.символів">
    <Statements>
     <ConditionalBlock>
      <If>
       <Condition>Not IsNull([Область])</Condition>
       <Statements>
        <ConditionalBlock>
         <If>
          <Condition>Not CheckOnlyUkranianSymbolsAndHyphen([Область])</Condition>
          <Statements>
           <Action Name="RaiseError">
            <Argument Name="Number">4</Argument>
            <Argument Name="Description">Область може містити лише українські літери або знак "-"</Argument>
           </Action>
          </Statements>
         </If>
        </ConditionalBlock>
       </Statements>
      </If>
     </ConditionalBlock>
     <ConditionalBlock>
      <If>
       <Condition>Not IsNull([Район])</Condition>
       <Statements>
        <ConditionalBlock>
         <If>
          <Condition>Not CheckOnlyUkranianSymbolsAndHyphen([Район])</Condition>
          <Statements>
           <Action Name="RaiseError">
            <Argument Name="Number">5</Argument>
            <Argument Name="Description">Район може містити лише українські літери або знак "-"</Argument>
           </Action>
          </Statements>
```

```
        </If>
       </ConditionalBlock>
      </Statements>
     </If>
    </ConditionalBlock>
    <ConditionalBlock>
     <If>
      <Condition>Not IsNull([Місто])</Condition>
      <Statements>
       <ConditionalBlock>
        <If>
         <Condition>Not
CheckOnlyUkranianSymbolsAndHyphenAndSpace([Місто])</Condition>
         <Statements>
          <Action Name="RaiseError">
           <Argument Name="Number">6</Argument>
           <Argument  Name="Description">Місто  може  містити  лише
українські літери або знак "-" або прогалину " "</Argument>
          </Action>
         </Statements>
        </If>
       </ConditionalBlock>
      </Statements>
     </If>
    </ConditionalBlock>
    <ConditionalBlock>
     <If>
      <Condition>Not IsNull([Вулиця])</Condition>
      <Statements>
       <ConditionalBlock>
        <If>
         <Condition>Not
CheckOnlyUkranianSymbolsAndHyphenAndSpace([Вулиця])</Condition>
         <Statements>
          <Action Name="RaiseError">
           <Argument Name="Number">7</Argument>
           <Argument  Name="Description">Вулиця  може  містити  лише
українські літери або знак "-"</Argument>
          </Action>
         </Statements>
        </If>
       </ConditionalBlock>
      </Statements>
     </If>
    </ConditionalBlock>
   </Statements>
  </StatementGroup>
  <StatementGroup Description="Форматування рядку">
   <Statements>
    <Action Name="SetField">
     <Argument Name="Field">Область</Argument>
     <Argument
Name="Value">Trim(NormalString([Область]))</Argument>
```

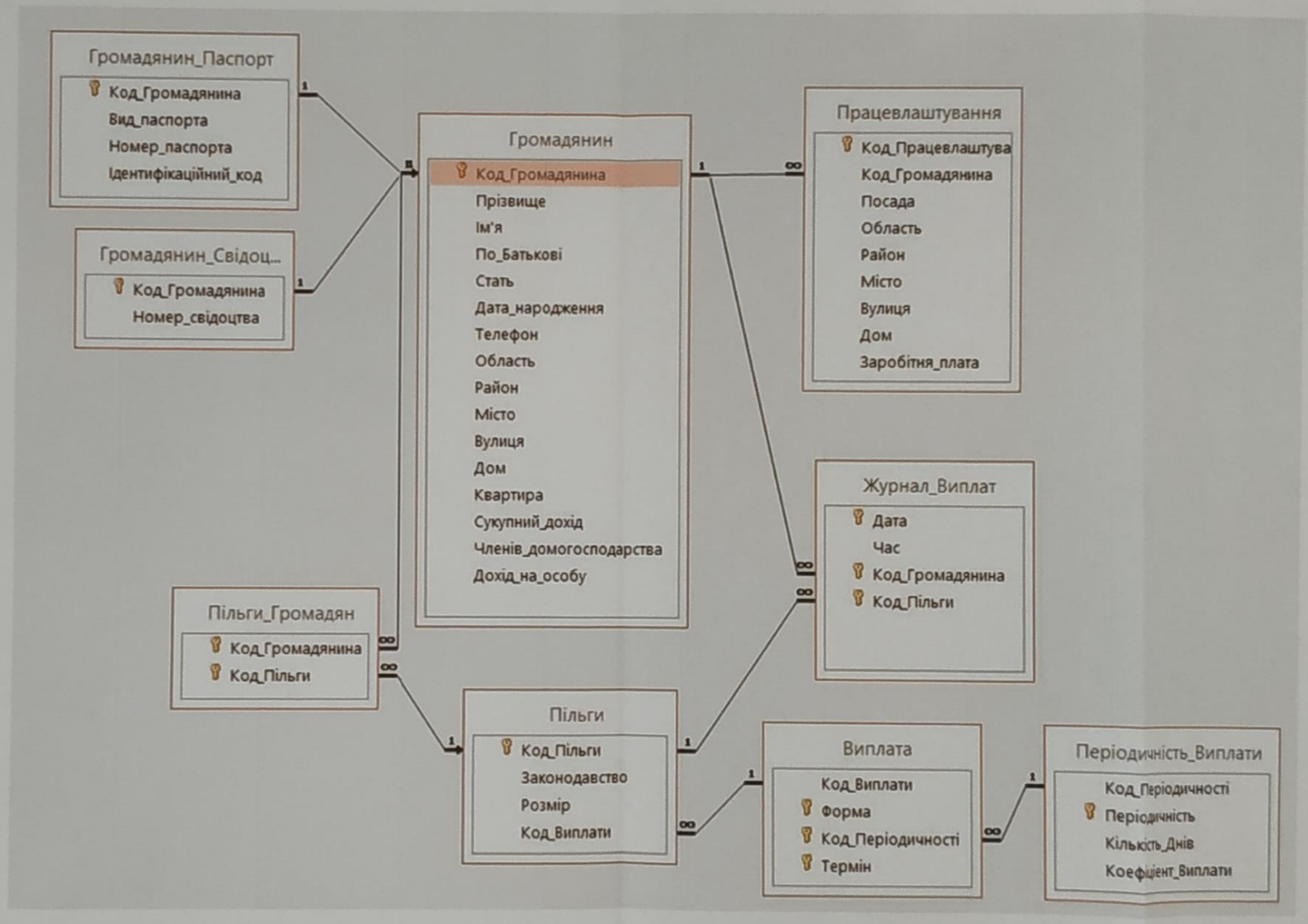
```
      </Action>
      <Action Name="SetField">
       <Argument Name="Field">Район</Argument>
       <Argument Name="Value">Trim(NormalString([Район]))</Argument>
      </Action>
      <Action Name="SetField">
       <Argument Name="Field">Місто</Argument>
       <Argument Name="Value">Trim(NormalString([Місто]))</Argument>
      </Action>
      <Action Name="SetField">
       <Argument Name="Field">Вулиця</Argument>
       <Argument Name="Value">Trim(NormalString([Вулиця]))</Argument>
      </Action>
     </Statements>
    </StatementGroup>
   </Statements>
  </DataMacro>
</DataMacros>
```

КОНЦЕПТУАЛЬНА МОДЕЛЬ ДАНИХ СИСТЕМИ



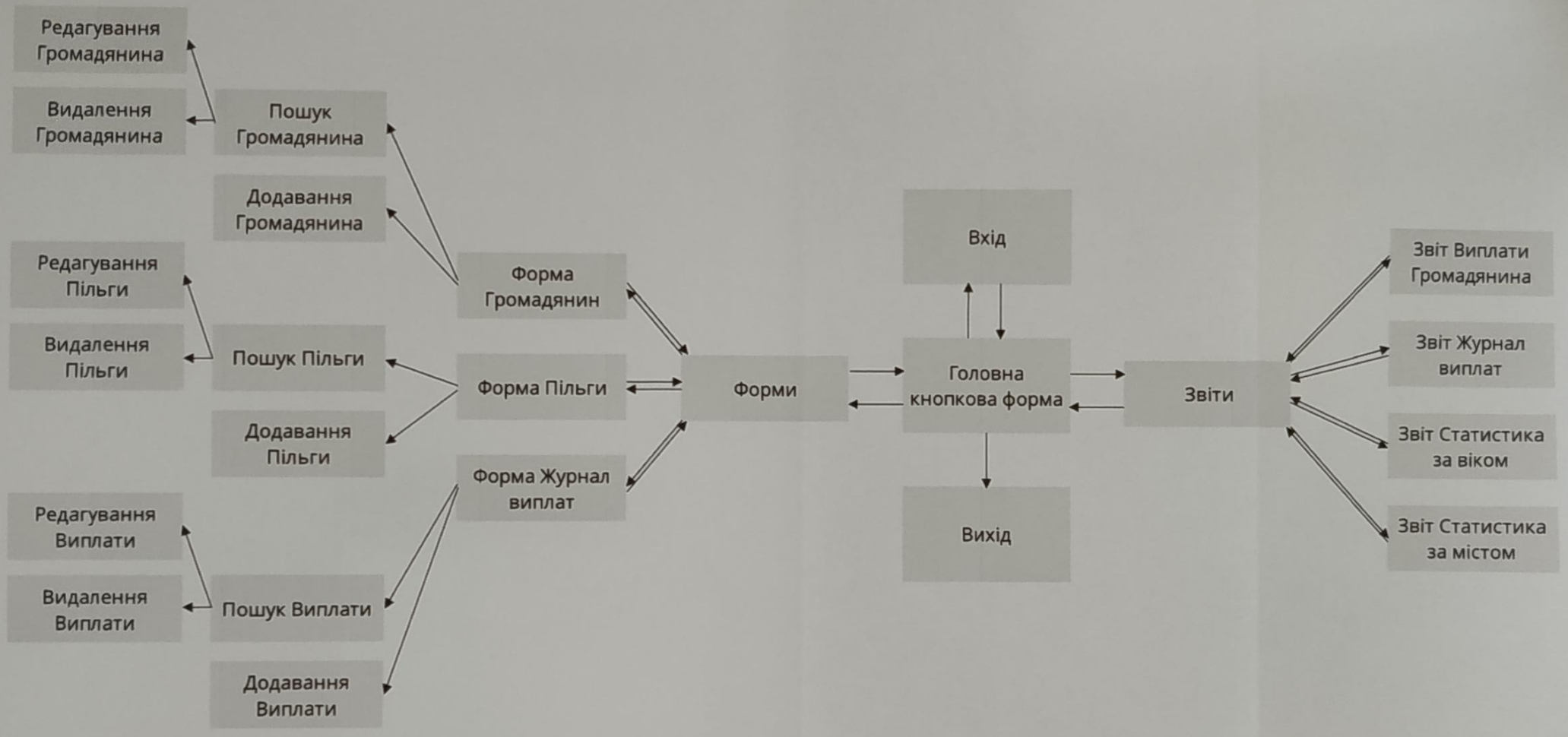
					13.02070849.00021 ПЛ1	
Зам.	Лист.	№ докум.	Підп.	Дата	Розробка інформаційної системи районного управління соціального захисту населення Концептуальна модель даних системи	Літ. / Маса / Масшт.
Розроб.		Шатов О.В.				
Перев.		Проскурін М.П.				Лист 1 / Листів 1
Т.контр.						
Н.контр.		Щербак Н.В.				НУ «Запорізька політехніка» КНТ-517
Затв.		Кудермєтов Р.К.				

СХЕМА ДАНИХ БАЗИ ДАНИХ ДОДАТКУ



Зам.	Лист.	№ докум.	Підп.	Дата	13.02070849.00021 ПЛ2		
Розроб.		Шатов О.В.			Розробка інформаційної системи районного управління соціального захисту населення Схема даних бази даних додатку	Лит.	Маса / Масшт.
Перев.		Проскурін М.П.				Лист 1	Листів 1
Т.контр.							
Н.контр.		Щербак Н.В.				НУ «Запорізька політехніка» КНТ-517	
Затв.		Кудерметов Р.К.					

ФУНКЦІОНАЛЬНА СХЕМА ДОДАТКУ



					13.02070849.00021 ПЛЗ		
Зам.	Лист.	№ докум.	Підп.	Дата	Розробка інформаційної системи районного управління соціального захисту населення Функціональна схема додатку	Лит.	Маса / Масшт.
Розроб.		Шатов О.В.					
Перев.		Проскурін М.П.				Лист 1	Листів 1
Т.контр.							
Н.контр.		Щербак Н.В.				НУ «Запорізька політехніка» КНТ-517	
Затв.		Кудерметов Р.К.					

ІНТЕРФЕЙС КОРИСТУВАЧА ДОДАТКУ

Пільги

Код пільги: 12

Законодавство: Учасник бойових дій

Розмір: 75.00 %

Код виплати: 10

Додати запис | Видалити запис

Виплати за віком

Прізвище	Ім'я	По батькові	Дата народження	Дата	Час	Законодавство	Розмір
Вік 21							
Шатов	Олег	Вікторович	30.08.2000	28.05.2021	18:41	Особа ЧАЕС - потерпілий	50.00%
						Усього	50.00%
						Кількість виплат 1	
Вік 25							
Микитюк	Генадій	Олександров	01.04.1996	24.03.2021	16:54	Особа ЧАЕС - потерпілий	50.00%
Микитюк	Генадій	Олександров	01.04.1996	08.05.2021	16:25	Учасник бойових дій	75.00%
						Усього	125.00%
						Кількість виплат 2	
Вік 31							
Захарчук	Поліна	Миколаївна	14.09.1990	10.02.2021	18:34	Багатодітна сім'я	50.00%
Захарчук	Поліна	Миколаївна	14.09.1990	28.05.2021	16:07	Особа ЧАЕС - ліквідатор	120.00%
						Усього	170.00%
						Кількість виплат 2	

ІКС соц. захисту населення

- Форми
- Звіти
- Вийти

Зам.	Лист.	№ докум.	Підп.	Дата	13.02070849.00021 ПЛ4		
Розроб.		Шатов О.В.			Розробка інформаційної системи районного управління соціального захисту населення Інтерфейс користувача додатку	Лит.	Маса	Масшт.
Перев.		Проскурін М.П.						
Т.контр.						Лист 1	Листів 1	
Н.контр.		Щербак Н.В.				НУ «Запорізька політехніка» КНТ-517		
Затв.		Кудерметов Р.К.						