МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І
ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

ОСНОВИ ОРГАНІЗАЦІЇ БАЗ ДАНИХ

НАВЧАЛЬНИЙ ПОСІБНИК

ДО ВИВЧЕННЯ ДИСЦИПЛІНИ «ОРГАНІЗАЦІЯ БАЗ ДАНИХ»

для студентів, що навчаються за спеціальностями
галузі 12- «Інформаційні технології»

Київ 2017

Навчальний посібник містить курс лекцій та лабораторний практикум з дисципліни «Організація баз даних», який відповідає базовому рівню знань у цьому напрямку.

Рекомендовано до друку Вченою радою Національного університету біоресурсів та природокористування України (протокол № __ від __.11.2017 р.)

Укладачі:

завідувач кафедри комп'ютерних наук НУБіП України, к.т.н., доцент Голуб Белла Львівна, асистент кафедри комп'ютерних наук НУБіП України Ящук Дар'я Юріївна

Рецензенти:

завідувач кафедри міжнародних перевезень та митного контролю Національного транспортного університету МОН України, академік Української академії економічної кібернетики, д.т.н., професор **Прокудін Г.С.**; провідний науковий співробітник Інституту напівпровідників ім. В.Є. Лошкарєва НАН України, д.т.н., **Бушма О.В.;** доцент кафедри автоматики та робототехнічних систем НУБіП України**,** к.т.н., доцент **Дудник А.О.**

Для студентів, що навчаються за спеціальностями галузі 12 – «Інформаційні технології» ОС «Бакалавр» та інших, які передбачають вивчення дисципліни «Організація баз даних». Посібник може бути використаний при організації навчальної практики студентів та для самостійного вивчення баз даних.

ЗМІСТ

ВСТУП

Студенти, що здобувають спеціальності в галузі інформаційних технологій, отримують знання з багатьох дисциплін, як фундаментального, так і професійно-орієнтовного характеру. Дисципліни професійно-орієнтовного характеру, у свою чергу, можна поділити на декілька напрямків, основний з яких – розробка і супроводження баз даних. Саме бази даних – це структурований і керований спосіб організації інформації при обробці її за допомогою комп'ютерної системи. Знання з основ організації баз даних надають можливість майбутньому спеціалісту відчувати себе впевнено при розробці систем з автоматизованою обробкою інформації, яких дуже багато: автоматизовані робочі місця певних спеціалістів; інформаційні системи, які охоплюють автоматизацію обробки інформації як у межах окремого підрозділу, так і у межах цілої організації; системи документообігу; автоматизовані системи управління технологічними процесами; веб-орієнтовані системи та багато інших. Натепер широкої популярності набули OLAP-технології, які відстежують процеси у режимі реального часу; технології Data Mining, які використовують методи штучного інтелекту для пошуку невідомих раніше закономірностей щодо даних; методи хмарних обчислень та обробки, так званих, великих даних (Big Data). І усі перераховані технології та методи використовують як джерела інформації бази даних. Більш того, саме завдячуючи широкому розповсюдженню баз даних і довгостроковому їх використанню, з'явилася можливість виникнення цих технологій.

Виходячи з наведеного, набуття професійних навичок розробки та управління базами даних є вкрай необхідним для фахівців з інформаційних технологій. Розгляду цих питань присвячена дисципліна «Організація баз даних» або «Бази даних». У межах цієї дисципліни

розглядається чимало питань, пов’язаних з розробкою і супроводженням баз даних. Основні питання такі:

- теорія баз даних,
- проектування об’єктів та супроводження бази даних у середовищі СУБД реляційного типу,
- основи мови SQL та принципи її використання.

Метою створення навчального посібника є представлення цих питань у доступній формі. Навчальний посібник розрахований на початковий рівень. Для спрощення засвоєння інструментарію розробки баз даних використовується система управління базами даних MS Access, що дозволяє початківцю у цьому предметі відразу приступати до розробки бази даних.

Навчальний посібник містить як теоретичний матеріал, так і лабораторний практикум. Теоретичний матеріал містить не лише суто теорію, а і відповідні приклади. У межах лабораторного практикуму розроблюється учбовий проект. Це дозволяє студенту крок за кроком пройти усі необхідні етапи розробки системи, які пов’язані з використанням бази даних, що, у свою чергу, допоможе йому краще засвоїти матеріал та набути практичний досвід із зазначених питань.

Навчальний посібник складається із шістьох розділів. У першому розділі розглядаються основні питання з теорії баз даних, такі як: поняття бази даних та систем управління базами даних, історія розвитку баз даних, основні концепції реляційної моделі, правила доктора Кодда, теорія нормалізації баз даних. У розділах з другого по четвертий представлена інформація щодо проектування і впровадження бази даних у середовищі MS Access, а саме: проектування і створення таблиць, форм, запитів та звітів. У п’ятому розділі розглядається мова SQL, основна увага в якому приділена команді SELECT, починаючи з простої форми цієї команди і завершуючи вкладеними запитами. Кожний із цих

розділів завершується контрольними питаннями, які студент може використати для самоперевірки. У шостому розділі у вигляді лабораторного практикуму наведений приклад розробки проекту з експертизи сортів рослин України, а також визначені завдання до кожної лабораторної роботи. Лабораторна робота ототожнює собою окремий етап створення як бази даних, так і завершеного проекту засобами MS Access.

Автори навчального посібника мають великий досвід, як педагогічний щодо викладання дисципліни з організації баз даних, так і професійний щодо розробки програмних проектів, основаних на використанні бази даних. Автором перших п’яти розділів є Голуб Б.Л., автором шостого розділу – Ящук Д.Ю.

1 ВВЕДЕННЯ В БАЗИ ДАНИХ

1.1 Загальні уявлення

База даних може бути визначена як деяка сукупність логічно зв'язаних даних та описів цих даних. База даних – це єдине велике вмістилище даних, яке один раз визначається, а потім використовується одночасно багатьма користувачами з багатьох підрозділів. База даних зберігає не тільки самі дані, а також і описи цих даних. У сукупності описи даних називаються системним каталогом або словником даних, а окремий елемент словника – метаданим, тобто даним, який описує дані. Саме наявність метаданих і забезпечує незалежність програм, які забезпечують роботу з базою даних, від самих даних.

Робота з базою даних здійснюється за допомогою системи керування або управління базами даних – СУБД. СУБД – це програмне забезпечення, за допомогою якого користувачі можуть визначати, створювати та підтримувати базу даних і здійснювати контролюємий доступ до неї.

Визначення бази даних здійснюється за допомогою мови визначення даних, яка дає можливість користувачеві вказати тип даних та їх структуру, а також певні обмеження на інформацію, що зберігається в базі даних.

Введення, редагування, видалення та пошук даних, як правило, здійснюється за допомогою мови керування даними.

Для забезпечення контролюємого доступу до бази даних СУБД повинна включати:

- систему забезпечення безпеки, яка забороняє несанкціонований доступ до інформації;
- систему підтримки цілісності даних, яка забезпечує не заперечливий стан даних бази даних;

- систему керування паралельною роботою з базою даних;
- систему відновлення бази даних, яка дозволяє відновити стан бази даних до моменту, коли відбувалися негаразди з апаратним або програмним забезпеченням.

Бази даних використовуються дуже широко в повсякденному житті. Наприклад, доступ до бази даних необхідний при здійсненні купівлі в супермаркеті, коли касир зчитує штрих-код із придбаного товару. Цей штрих-код використовується для пошуку ціни конкретного товару в базі даних усіх товарів. Далі програма відніме кількість придбаного товару зі складу товарів супермаркету і висвітить на касовому апарату його коштовність. Якщо кількість товару на складі стане меншою, ніж заздалегідь визначений рівень, програма зможе автоматично відправити заявку на придбання додаткової кількості цього товару.

Інший приклад – придбання квитків на літак або потяг. За запитом пасажира продавець квитків продивляється базу даних, де відображуються розклад рейсів та наявність квитків. Програма дозволить придбати квиток і автоматично відредагує інформацію щодо наявності вільних місць на літаку або потягу.

В університеті також може існувати база даних студентів, в якій представлена інформація щодо здачі екзаменів, наявності стипендії, заборгованості за різними предметами тощо.

1.2 Компоненти середовища СУБД

Середовище СУБД складається з п'яти компонентів: апаратне забезпечення, програмне забезпечення, дані, процедури та користувачі. Це представлено на рис.1.1.

Апаратне забезпечення може бути представлено як одним комп'ютером, так і комп'ютерною мережею. Апаратне забезпечення

залежить від організації, для якої розроблена база даних, та конкретною СУБД. Деякі СУБД розраховані задля роботи з вибірковими комп'ютерами та операційними системами, інші розраховані на велике коло апаратного забезпечення та операційних систем.

Програмне забезпечення охоплює як програмне забезпечення самої СУБД, так і прикладне програмне забезпечення, операційну систему та програмне забезпечення комп'ютерної мережі, якщо СУБД використовується в мережі. Прикладне програмне забезпечення може розроблюватися як на мовах високого рівня, таких, як C, COBOL, FORTRAN, Pascal, так і на спеціальних мовах, призначених саме для роботи з базами даних, наприклад, SQL. СУБД можуть мати і внутрішні можливості для створення прикладного програмного забезпечення.

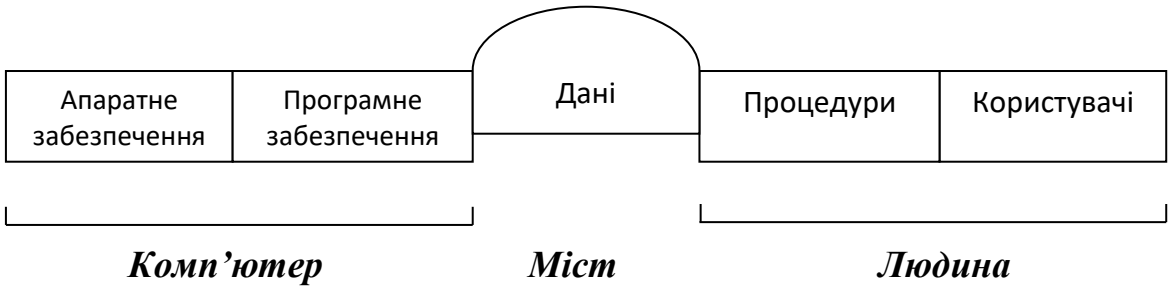


Рис.1.1 Середовище СУБД

Дані – найважливіша компонента СУБД. На малюнку дані зображені у вигляді моста, який з'єднує комп'ютер і людину. База даних містить як робочі дані, так і описи цих даних – метадані. Структура бази даних називається схемою бази даних.

К процедурам відносяться інструкції та правила, які повинні враховуватися під час проектування та використання бази даних:

- реєстрація в СУБД;
- використання прикладного програмного забезпечення;
- запуск і зупинка СУБД;
- створення резервних копій;
- відновлення бази даних після можливих відмов системи тощо.

Щодо останньої п'ятої компоненти, то тут можна виділити чотири групи користувачів системи:

1) адміністратор бази даних – відповідає за всю систему в цілому, за її безпеку, за апаратне та програмне забезпечення тощо;

2) розробник бази даних – займається розробкою логічної структури бази даних та реалізацією логічної структури на фізичному рівні, вибором конкретних структур збереження даних та методів доступу до них, проектує засоби захисту даних;

3) прикладний програміст – займається розробкою додатків до бази даних, які і забезпечують інформаційні потреби кінцевих користувачів;

4) кінцевий користувач – клієнт бази даних, для яких вона створюється і підтримується.

1.3 Історія розвитку СУБД

Вважається, що розвиток СУБД розпочався на початку 60-х років минулого сторіччя, коли розроблювався проект польоту APOLLO на місяць. На той час не було ніяких систем, які б були здатні оброблювати та керувати великими об'ємами даних. В результаті спеціалістами було розроблено спеціальне програмне забезпечення під назвою GUAM, яке дозволяло об'єднувати маленькі компоненти, які є частинами більших компонентів, до тих пір, поки не створюється повний проект. Така структура бази даних отримала назву ієрархічної структури.

Трохи пізніше була розроблена система під назвою IDS, яка поклала початок розробці систем із мережевою структурою. Мережева СУБД використовувалась для баз даних більш складної структури, ніж ієрархічна.

У 1965 р. з'явився перший стандарт Асоціації по мовам систем обробки даних – Conference of Data System Languages (CODASYL), який

визначив ряд фундаментальних понять в теорії систем баз даних. Ці поняття і досі є основними для мережевої моделі даних.

Однак цим двом першим моделям притаманні наведені нижче недоліки.

- ✓ Навіть для виконання простих запитів з використанням переходів і доступом до певних записів необхідно створювати досить складні програми.
- ✓ Незалежність від даних існує лише в мінімальному обсязі.
- ✓ Відсутні теоретичні основи.

У подальший розвиток теорії баз даних великий внесок був зроблений американським математиком Е.Ф.Коддом. У 1970 році Е.Ф. Кодд, який працював в корпорації IBM, опублікував статтю з реляційної моделі даних, що дозволяла усунути недоліки своїх попередниць – мережевої і ієрархічної моделей даних.

Слідом за цим з'явилася безліч експериментальних реляційних СУБД, а перші комерційні продукти з'явилися в кінці 70-х – початку 80-х років. Особливо слід відзначити проект System R, розроблений в корпорації IBM в кінці 70-х років (Astrahan et al., 1976). Цей проект був задуманий з метою довести практичність реляційної моделі, що досягалося за допомогою реалізації передбачених нею структур даних і необхідних функціональних можливостей. На основі цього проекту були отримані найважливіші результати, а саме:

• була розроблена структурована мова запитів SQL, яка з тих пір стала стандартною мовою будь-яких реляційних СУБД;

• у 80-х роках були створені різні комерційні реляційні СУБД, наприклад, DB2 або SQL/DS корпорації IBM, Oracle корпорації Oracle, ін.

Однак реляційна модель також має деякі недоліки, зокрема, обмежені можливості моделювання. Для вирішення цієї проблеми було

виконано великий обсяг дослідницької роботи. У 1976 році Чен запропонував модель "сутність-зв'язок" (Entity-Relationship model – ER-модель), яка в даний час стала основою методології концептуального проектування баз даних та методології логічного проектування реляційних баз даних. У 1979 році Кодд зробив спробу усунути недоліки власної основної роботи і опублікував розширену версію реляційної моделі – RM/T (1979), потім ще одну версію – RM / V2 (1990). Спроби створення моделі даних, що дозволяє більш точно описувати реальний світ, називають семантичним моделюванням даних (semantic data modeling).

У відповідь на все зростаючу складність додатків баз даних з'явилися дві нові системи: об'єктно-орієнтовані СУБД, або ОО СУБД (Object-Oriented DBMS - OODBMS), і об'єктно-реляційні СУБД, або ОР СУБД (Object-Relational DBMS - ORDBMS).

Можна виділити чотири етапи в розвитку цього напрямку в обробці даних. Однак необхідно зауважити, що все-таки немає жорстких часових обмежень в цих етапах: вони плавно переходять один в інший і навіть співіснують паралельно.

І етап – бази даних на великих ЕОМ.

• Організація баз даних на великих машинах типу IBM 360/370, ЄС-ЕОМ і міні-ЕОМ типу PDP11 (фірми Digital Equipment Corporation - DEC), різних моделях HP (фірми Hewlett Packard).

• Всі СУБД базувалися на потужних мультипрограмних операційних системах (MVS, SVM, RTE. OSRV, RSX, UNIX), тому в основному підтримувалася робота з централізованою базою даних в режимі розподіленого доступу (база даних була централізованою, зберігалася на пристроях зовнішньої пам'яті однієї центральної ЕОМ, а доступ до неї здійснювався від багатьох користувачів – завдань).

• Функції управління розподілом ресурсів в основному здійснювала операційна система.

• Значна роль відводилася адмініструванню даних.

• Підтримувалися мови низького рівня маніпулювання даними, орієнтовані на навігаційні методи доступу до даних.

II етап розвитку СУБД пов'язаний з появою персональних комп'ютерів і, так званих, настільних СУБД.

Особливості другого етапу такі.

• Всі СУБД були розраховані на створення БД в основному з монопольним доступом.

• Більшість СУБД мали розвинений і зручний призначений для користувача інтерфейс.

• Більшість СУБД пропонували розвинений і зручний інструментарій для розробки готових додатків без програмування.

• У всіх настільних СУБД підтримувався тільки зовнішній рівень представлення реляційної моделі, тобто тільки зовнішній табличний вигляд структур даних.

• У настільних СУБД були відсутні засоби підтримки посилальної і структурної цілісності бази даних.

• При наявності високорівневих мов маніпулювання даними типу реляційної алгебри і SQL у настільних СУБД підтримувалися низькорівневі мови маніпулювання даними на рівні окремих рядків таблиць.

• Наявність монопольного режиму роботи фактично призвело до виродження функцій адміністрування БД і до відсутності розвинутих інструментальних засобів адміністрування БД.

• Порівняно скромні вимоги до апаратного забезпечення з боку настільних СУБД.

Яскраві представники цього сімейства - Dbase, FoxPro, Clipper, Paradox.

III етап - розподілені бази даних.

Після процесу "персоналізації" почався зворотний процес – інтеграція. Зі збільшенням кількості локальних мереж, все більше інформації стало передаватися між комп'ютерами. Гостро постало питання узгодженості даних, що зберігаються і обробляються в різних місцях, але логічно один з одним пов'язаних. Успішне вирішення цих завдань призводить до появи розподілених баз даних, що зберігають усі переваги настільних СУБД і, в той же час, дозволяють організовувати паралельну обробку інформації і підтримку цілісності баз даних.

Особливості третього етапу такі.

• Підтримка повної реляційної моделі, а саме:

- структурної цілісності – допустимими є тільки дані, представлені у вигляді відносин реляційної моделі;
- мовної цілісності, тобто мов маніпулювання даними високого рівня (в основному SQL);
- посилальної цілісності, контролю за дотриманням посилальної цілісності протягом всього часу функціонування системи, і гарантій неможливості з боку СУБД порушити ці обмеження.

• Більшість сучасних СУБД розраховані на набір крос-платформних інструментів (комп'ютери з різною архітектурою, різною операційною системою, при цьому для користувача доступ до даних, керованих СУБД на різних платформах, практично не розрізняється).

• Необхідність підтримки багатокористувацької роботи з базою даних і можливість децентралізованого зберігання даних з реалізацією загальної концепції засобів захисту даних і як наслідок розвиток засобів адміністрування БД.

Представниками СУБД, що належать до цього етапу, можна вважати MS Access 2007 і всі сучасні сервери баз даних Oracle 11g, MS SQL Server, System 11, Informix та ін.

IV етап

Цей етап характеризується появою нової технології доступу до даних – Інтранет. Основна відмінність цього підходу від технології клієнт-сервер полягає в тому, що відпадає необхідність використання спеціалізованого клієнтського програмного забезпечення. Для роботи з віддаленою базою даних використовується стандартний браузер Інтернету, наприклад Microsoft Internet Explorer. Зручність цього підходу призвело до того, що він став використовуватися не лише для віддаленого доступу до баз даних, але і для користувачів локальної мережі підприємства.

Представник цього сімейства: MySQL.

1.4 Основні концепції реляційної моделі

У будь-якій реляційній базі даних користувач сприймає базу даних як деяку сукупність таблиць. Кожна таблиця в реляційній моделі називається «*відношення*». Відношення має вигляд двомірної таблиці, рядок якої – окремий «*запис*», а стовпчик з назвою – «*атрибут*».

Наприклад, інформація щодо студентів однієї групи може бути представлена відношенням Група з атрибутами номер за порядком, ідентифікаційний код, прізвище, ім'я, по батькові, дата народження, місце народження.

Сукупність значень, на якому визначаються атрибути, називається «*домен*». Іншими словами, домен визначає, які значення і якого типу можуть бути введені для атрибутів. Наприклад, домен для атрибута дата народження може бути визначений як дата в діапазоні від 01.01.1975 року.

Кількість атрибутів у відношенні отримала назву «*ступінь відношення»*. У нашому прикладі ступінь відношення Група становить 7.

Кожний рядок відношення, тобто кожний рядок таблиці, має назву «*кортеж»*. Кортежі можуть бути записані в довільному порядку у відношенні, але від цього структура відношення не змінюється.

Кількість кортежів у відношенні має назву «*кардинальність»*.

Таким чином, альтернативна термінологія може бути представлена так:

відношення – таблиця;

атрибут – стовпчик – поле;

кортеж – рядок – запис.

1.5 Правила доктора Кодда

Доктор Кодд створив ці правила як частину своєї кампанії по запобіганню розмиванню його бачення реляційності через те, що продавці систем керування базами даних на початку 1980х просто видавали свої старі продукти за реляційні розробки. Насправді, правила настільки суворі, що всі популярні так звані «реляційні» СУБД не відповідають багатьом критеріям. Особливо складні 6, 9, 10, 11 і 12.

0. ***Фундаментальне правило*** (Foundation Rule)

Реляційна СУБД повинна бути здатна повністю керувати базою даних, використовуючи зв'язки між даними.

1. ***Інформаційне правило*** (Information Rule)

Інформація повинна бути представлена у вигляді даних, що зберігаються в осередках. Дані, що зберігаються в комірках, повинні бути атомарним. Порядок рядків в реляційній таблиці не повинен впливати на зміст даних.

Правило інформації. Вся інформація в базі даних повинна бути представлена виключно на логічному рівні і лише одним способом – у

вигляді значень, що містяться в таблицях. Фактично це неформальне визначення реляційної бази даних.

2. ***Правило гарантованого доступу*** (Guaranteed Access Rule)

Доступ до даних повинен бути вільним від двозначності. До кожного елементу даних повинен бути гарантований доступ за допомогою комбінації імені таблиці, первинного ключа рядку й імені стовпця.

Правило 2 вказує на роль первинного ключа при пошуку інформації в базі даних. Ім'я таблиці дозволяє знайти необхідну таблицю, ім'я стовпця дозволяє знайти потрібний стовпець, а первинний ключ дозволяє знайти рядок, що містить шуканий елемент даних.

3. ***Систематична обробка Null-значень*** (Systematic Treatment of Null Values)

Невідомі значення NULL, відмінні від будь-якого відомого значення, повинні підтримуватися для всіх типів даних при виконанні будь-яких операцій. Наприклад, для числових даних невідомі значення не повинні розглядатися як нулі, а для символьних даних - як порожні рядки.

Правило 3 вимагає, щоб відсутні дані можна було уявити за допомогою недійсних значень (NULL).

4. ***Правило доступу до системного каталогу на основі реляційної моделі*** (Dynamic On-line Catalog Based on the Relational Model)

Словник даних повинен зберігатися у формі реляційних таблиць, і СУБД повинна підтримувати доступ до нього за допомогою стандартних мовних засобів, тих самих, які використовуються для роботи з реляційними таблицями, що містять дані користувача.

Правило 4 свідчить, що реляційна база даних повинна сама себе описувати. Іншими словами, база даних повинна містити набір системних таблиць, що описують структуру самої бази даних.

5. ***Правило повноти підмови маніпулювання даними*** (Comprehensive Data Sublanguage Rule)

Система управління реляційними базами даних повинна підтримувати хоча б одну реляційну мову, яка

а) має лінійний синтаксис,

б) може використовуватися як інтерактивною, так і в прикладних програмах,

в) підтримує операції визначення даних, визначення уявлень, маніпулювання даними (інтерактивні та програмні), обмежувачі цілісності, управління доступом та операції управління транзакціями (begin, commit і rollback).

Правило 5 вимагає, щоб СУБД використовувала мову реляційної бази даних. Така мова повинна підтримувати всі основні функції СУБД - створення бази даних, читання і введення даних, реалізацію захисту бази даних і т.д.

6. ***Правило модифікації представлень*** (View Updating Rule)

Кожне подання має підтримувати всі операції маніпулювання даними, які підтримують реляційні таблиці: операції вибірки, вставки, модифікації і видалення даних.

Правило 6 стосується уявлень, які є віртуальними таблицями, які дозволяють показувати різним користувачам різні фрагменти структури бази даних.

7. ***Правило високорівневих операцій модифікації даних*** (High-level Insert, Update, and Delete)

Операції вставки, модифікації і видалення даних повинні підтримуватися не тільки по відношенню до одного рядку реляційної таблиці, але по відношенню до будь-якої безлічі рядків.

Правило 7 акцентує увагу на тому, що бази даних за своєю природою орієнтовані на множини. Воно вимагає, щоб операції додавання, видалення і оновлення можна було виконувати над множинами рядків.

8. ***Правило фізичної незалежності даних*** (Physical Data Independence)

Додатки не повинні залежати від використовуваних способів зберігання даних на носіях, від апаратного забезпечення комп'ютерів, на яких знаходиться реляційна база даних.

Прикладні програми і утиліти для роботи з даними повинні на логічному рівні залишатися недоторканими за будь-яких змінах способів зберігання даних або методів доступу до них.

9. ***Правило логічної незалежності даних*** (Logical Data Independence)

Представлення даних в додатку не повинно залежати від структури реляційних таблиць. Якщо в процесі нормалізації одна реляційна таблиця розділяється на дві, подання повинне забезпечити об'єднання цих даних, щоб зміна структури реляційних таблиць не позначалося на роботі додатків.

Правила 8 і 9 означають відділення користувача та прикладної програми від низькорівневої реалізації бази даних.

10. ***Правило незалежності контролю цілісності*** (Integrity Independence)

Вся інформація, необхідна для підтримки цілісності, повинна бути в словнику даних. Мова для роботи з даними повинна виконувати перевірку вхідних даних і автоматично підтримувати цілісність даних.

Правило 10 говорить, що мова бази даних повинен підтримувати обмежувальні умови, що накладаються на дані, що вводяться і дії, які можуть бути виконані над даними.

11. ***Правило незалежності від розміщення*** (Distribution Independence)

База даних може бути розподіленої, може перебувати на декількох комп'ютерах, і це не повинно впливати на додатки. Перенесення бази даних на інший комп'ютер не повинен впливати на додатки.

Правило 11 говорить, що мова бази даних повинна забезпечувати можливість роботи з розподіленими даними, розташованими на інших комп'ютерних системах.

12. ***Правило узгодженості мовних рівнів*** (The Nonsubversion Rule)

Якщо використовується низькорівнева мова доступу до даних, він не повинен ігнорувати правила безпеки і правила цілісності, які підтримуються мовою більш високого рівня.

Правило 12 запобігає використанню інших можливостей для роботи з базою даних крім мови бази даних, оскільки це може порушити її цілісність.

1.6 Нормалізація даних

Головний постулат БД – дані не повинні бути надлишковими, тобто необхідно зводити до мінімуму повторення даних.

Нормалізація – це процес скорочення повторень інформації в базі даних. Нормалізуються в БД не тільки дані, а і імена.

З іншого боку, ненормалізована БД – це база, яка не розділена на менші, логічно єдині і більш керовані таблиці.

Розрізняють декілька етапів процесу нормалізації. В результаті виконання кожного з них створюється, так звана, *нормальна форма*.

Говорять, що нормальна форма – це міра глибини, до якої повинна бути виконана нормалізація БД.

Основні властивості нормальних форм:

- кожна наступна нормальна форма в деякому розумінні краще попередньої;
- при переході до наступної нормальної форми властивості попередніх нормальних форм зберігаються.

Перед проведенням процесу нормалізації дуже важливим є використання інформативних імен будь-яких одиниць даних.

Перша нормальна форма (1NF). Метою першої нормальної форми є розділення БД на логічні одиниці – таблиці. Після створення таблиць для більшості з них будуть визначені ключові поля. Іншими словами, для переходу до першої нормальної форми БД повинна бути розбита на декілька логічних одиниць, в кожній з них визначений ключ і відсутні групи, що повторюються.

Найбільш важливі на практиці нормальні форми відносин ґрунтуються на фундаментальному в теорії реляційних баз даних понятті функціональної залежності.

Для подальшого викладу нам будуть потрібні кілька визначень.

Визначення 1. *Функціональна залежність*

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

Визначення 2. *Повна функціональна залежність*

Функціональна залежність R.X (r) R.Y називається повною, якщо атрибут Y не залежить функціонально від будь-якої точної підмножини X.

Визначення 3. *Транзитивна функціональна залежність*

Функціональна залежність R.X -> R.Y називається транзитивною, якщо існує такий атрибут Z, що є функціональні залежності R.X -> R.Z і R.Z -> R.Y і відсутня функціональна залежність R.Z -> R.X. (При відсутності останньої вимоги ми мали б "нецікаві" транзитивні залежності в будь-якому відношенні, що володіє декількома ключами.)

Визначення 4. *Неключовий атрибут*

Неключовим атрибутом називається будь-який атрибут відносини, що не входить до складу первинного ключа.

Визначення 5. *Взаємно незалежні атрибути*

Два або більше атрибута взаємно незалежні, якщо жоден з цих атрибутів не є функціонально залежним від інших.

Друга нормальна форма (2NF). Відношення R знаходиться в другій нормальній формі (2NF) у тому і тільки у тому випадку, коли знаходиться в 1NF, і кожен неключовий атрибут повністю залежить від первинного ключа. Метою другої нормальної форми є відокремлення даних, які майже не залежать від ключового поля, і розміщення таких даних в окрему таблицю. Іншими словами, друга нормальна форма отримується з першої шляхом подальшого розділення таблиць на менші за розміром.

Третя нормальна форма (3NF). Відношення R знаходиться в третій нормальній формі (3NF) в тому і тільки в тому випадку, якщо знаходиться в 2NF і кожен неключовий атрибут не транзитивно залежить від первинного ключа. Метою третьої нормальної форми є видалення із таблиць даних, які не залежать від ключа.

У теорії реляційних баз даних за звичай виділяються ще такі нормальні форми:

- нормальна форма Бойса-Кодда (BCNF);
- четверта нормальна форма (4NF);

• п’ята нормальна форма, або нормальна форма проекції-об’єднання (5NF або PJ/NF).

Переваги нормалізації. Нормалізація має цілий ряд переваг:

- краща загальна організація БД;
- скорочення кількості непотрібних повторень БД;
- погодження даних усередині таблиць;
- більш гнучка структура БД;
- ефективні можливості забезпечення безпеки та надійності БД.

Процес нормалізації покращує роботу з базою усім – починаючи з адміністратора БД і закінчуючи кінцевим користувачем. Зменшується кількість повторень даних, що спрощує структуру даних і призводить до економії дискового простору. Завдячуючи скороченню дублювання даних, зменшується ймовірність їхньої не погодженості. Оскільки в процесі нормалізації даних БД розділяється на менші таблиці, змінювати невеликі таблиці стає простіше. Нарешті, підвищується безпека – нормалізація спрощує управління безпекою.

Недоліки нормалізації та процес денормалізації. Нормалізація має один суттєвий недолік – сповільнення роботи самої БД. Чим більше невеликих таблиць пов’язані між собою, тим довший шлях необхідно подолати для пошуку тої чи іншої інформації.

Денормалізація – це процес модифікації структури таблиць нормалізованої БД з метою підвищення продуктивності за рахунок дозволу деякого керованого надлишку даних. Денормалізація знижує рівень нормалізації на один-два рівні. Денормалізація передбачає об’єднання деяких таблиць та створення таблиць з даними, що дублюються, з метою зменшення кількості зв’язаних таблиць, що використовуються при пошуку даних. При цьому треба враховувати, що отримавши при збільшенні швидкості роботи БД, втрачається простота управління такими даними.

Контрольні питання

1. Які основні компоненти середовища СУБД Ви можете назвати?

2. Що означає компонент «Процедури» середовища СУБД?

3. Які основні моделі баз даних Ви знаєте?

4. Які основні етапи розвитку СУБД можна виділити?

5. Що таке реляційна БД та які її основні концепції?

6. Яке правило доктора Кодда є неформальним визначенням реляційної бази даних?

7. Поясніть процес нормалізації даних.

8. Що означає нормальна форма?

9. Чим відрізняються друга та третя нормальні форми?

10. Який недолік процесу нормалізації даних?

2 ПРОЕКТУВАННЯ БАЗИ ДАНИХ

2.1 Система управління реляційними базами даних Microsoft Access

Система управління реляційними базами даних (СУРБД) входить до складу пакета програм Microsoft Office, але може використовуватися як самостійний продукт.

Призначення СУРБД MS Access полягає в проектуванні, розробці додатків та супроводженні реляційної бази даних.

Основні етапи, які складають процес проектування бази даних, є такими:

1) визначення кількості таблиць та їх змістовного призначення;
2) визначення атрибутів у кожній таблиці;
3) визначення атрибутів, значення яких є унікальними усередині таблиці;
4) визначення зв'язків між таблицями.

Розробка додатків в СУРБД MS Access полягає в створенні форм введення інформації в кожну таблицю, запитів для формування механізму пошуку інформації в базі даних, звітів для отримання результатів пошуку в наочній формі, сторінок доступу до даних, макросів та модулів для автоматичного виконання деяких операцій.

Супроводження бази даних полягає в можливості введення, корегування, захисту даних у базі.

2.2 Створення бази даних

Створенню бази даних передує процес проектування бази даних, який здійснюється після ретельного аналізу предметної області, як правило, у вигляді логічної моделі даних. Реляційна база даних має відповідати певним вимогам, головна з яких – структура бази даних повинна бути представлена не менше, ніж у третій нормальній формі.

MS Access має файл-серверну архітектуру. Для користувачів це означає таке:

1) уся база даних розміщена в одному файлі, включаючи не лише таблиці, а і усі інші необхідні об’єкти: запити, звіти, макроси, модулі, сторінки доступу;

2) якщо відбувається запит з клієнтського місця до сервера, на якому розташована база даних, клієнт отримує увесь файл на свій комп’ютер, і наступний клієнт зможе працювати з даними лише у режимі перегляду, хоча також отримує на своє робоче місце увесь файл.

В СУРБД MS Access створення бази даних можна виконати двома способами. Перший спосіб полягає у створенні порожньої бази даних, а потім доповненні її таблицями, формами, звітами, запитами тощо. Другий спосіб полягає у використанні так званого майстра бази даних. Цей спосіб пропонує готові рішення щодо таблиць, їх структур, форм, звітів тощо. Перший спосіб є універсальним і використовується для будь-якої бази даних. Другий спосіб може бути використаний тоді, коли нова база даних відноситься до одного із запропонованих майстром бази даних типів.

Середовище MS Access змінювалося від версії до версії. Спробуємо узагальнити механізм створення нової бази даних незалежно від версії.

Створення бази даних без допомоги майстра

1. При запуску MS Access відкривається діалогове вікно, у якому пропонується створити нову базу даних чи відкрити існуючу. Необхідно у цьому діалоговому вікні вибрати перемикач «Нова база даних» і натиснути кнопку «ОК». Якщо база даних уже була відкрита чи якщо було закрито діалогове вікно, що з’являється при запуску MS Access, необхідно натиснути кнопку «Створити» на панелі інструментів і двічі клацнути значок «База даних» на вкладці «Загальне».

2. Вказавши ім'я і каталог бази даних, необхідно натиснути кнопку «Створити».

Після створення порожньої бази даних необхідно створити об'єкти цієї бази даних, а саме: таблиці, форми, запити, звіти, сторінки доступу, макроси і модулі.

2.3 Проектування таблиць

Основним об'єктом бази даних є таблиця. З точки зору СУРБД Microsoft Access таблиця — це набір даних за конкретною темою усередині предметної області (або набір атрибутів певної сутності предметної області). Використання окремої таблиці для кожної теми означає, що відповідні дані збережені тільки один раз. Це робить базу даних більш ефективною і знижує число помилок при введенні даних.

Перед тим, як починати проектувати таблицю, необхідно ознайомитися з двома важливими речами: типами даних і ключовим полем (первинним ключем).

Типи в Microsoft Access. При виборі типу даних необхідно враховувати таке:

• *Які значення* повинні відображатися в поле. Наприклад, не можна зберігати текст у поле, що має числовий тип даних.

• *Скільки місця* необхідно для збереження значень у поле.

• *Які операції* повинні виконуватися зі значеннями в полі. Наприклад, підсумовувати значення можна в числових полях і в полях, що мають грошовий формат, а значення в текстових полях і полях об'єктів OLE не можна.

• Чи потрібне *сортування чи індексування* атрибутів.

• Чи необхідно використання полів в *угрупованні* записів у запитах чи звітах.

• Яким чином повинні бути *відсортовані* значення в полі.

Зведення типів даних полів, доступних у Microsoft Access, представлені у таблиці 1.

Таблиця 1

Основні типи даних Microsoft Access

Тип даних	Використання	Розмір
Текстовий (в останніх версіях – *Короткий текст*)	Текст чи комбінація тексту і чисел. Наприклад, адреси, а також числа, що не вимагають обчислень. Наприклад, номера телефонів, інвентарні номери чи поштові індекси.	До 255 символів. MS Access зберігає тільки введені в поле символи; незаповнені частини полів типу «Текстовий» не зберігаються. Для визначення максимальної кількості символів, які можна ввести, використовується властивість «Розмір поля» («FieldSize»).
Поле Мето (в останніх версіях *Довгий текст*)	Довгий текст або числа, наприклад, примітки або описи	До 64 000 символів.
Числовий	Числові дані, які використовуються для математичних обчислень, за виключенням фінансових розрахунків (для них використовується тип	1, 2, 4 або 8 байтів. 16 байтів тільки для кодів реплікації. Для більш точного визначення типу числа використовується властивість «Розмір поля» («FieldSize»).

Тип даних	Використання	Розмір
	«Грошовий»).	
Дата/час	Дата і час	8 байтів
Грошовий	Значення валют. Грошовий тип використовується для запобігання округлень під час розрахунків. Припускає до 15 символів у цілій частині числа та 4 - в дробовій.	8 байтів
Лічильник	Автоматична вставка послідовних (збільшених на 1) або випадкових чисел при доповненні записів.	4 байти
Логічний	Поля, які містять тільки одне з двох можливих значень, таких як «Да/Нет», «Истина/Ложь», «Вкл/Выкл».	1 біт
Поле об'єкта OLE	Об'єкти (наприклад, документи Microsoft Word, електронні таблиці Microsoft Excel, рисунки, звуки тощо), які створені в інших	До 1 Гб

Тип даних	Використання	Розмір
	программах, з протоколом OLE.	
Гіперпосилання	Поле, в якому зберігаються гіперпосилання: шлях до файлу, електронна адреса тощо.	До 64000 символів
Майстер підстановок	Створює поле, яке дозволяє вибрати значення із іншої таблиці чи із списку значень, використовуючи поле із списком.	4 байти

Ключове поле (первинний ключ). Для ідентифікації кожного запису в таблиці використовується унікальний маркер, який називається *первинним ключем*. Як номерний знак ідентифікує автомобіль, так і первинний ключ однозначно визначає запис у таблиці. Якщо для таблиці визначений первинний ключ, то Microsoft Access запобігає дублювання ключа або введення значення Null в це поле. Ключ таблиці служить для посилання на записи таблиці з іншої таблиці.

В Microsoft Access можна виділити три типи ключових полів: лічильник, простий ключ та складений ключ.

Поле лічильника можна задати таким чином, щоб за доповненням кожного запису в таблицю в це поле автоматично заносилось порядкове число. Указівка такого поля у якості ключового є найбільш простим способом створення первинного ключа. Якщо до створення таблиці

ключові поля не були визначені, Microsoft Access пропонує створити ключове поле автоматично. При цьому буде створено ключове поле лічильника.

У Microsoft Access існує два способи створення таблиці. Для введення власних даних можна створити порожню таблицю. Можна також створити таблицю, використовуючи вже існуючі дані з іншого джерела.

Для створення таблиці в режимі конструктора необхідно виконати дії, що перераховані нижче.

1) Переключитися у вікно бази даних. Для переключення у вікно бази даних з інших вікон натиснути клавішу F11.

2) Вибрати значок «Таблиці» у списку «Об’єкти» і натиснути кнопку «Створити» на панелі інструментів вікна бази даних.

3) Вибрати «Режим конструктора». У цьому режимі з’явиться таблиця як на рис.2.1. Заповнюючи цю таблицю, розробник визначає усі необхідні атрибути та їхні характеристики.

4) Для цього в колонці «Им’я поля» задається назва атрибута, в колонці «Тип даних» – один з можливих типів кожного атрибута, в колонці «Опис» – коментар до кожного атрибута. Остання колонка є не обов'язковою.

5) До того, як зберегти таблицю, визначити первинний ключ.

Поле первинного ключа визначати не обов'язково, але бажано. Якщо первинний ключ не був визначений, Microsoft Access при збереженні таблиці запитає, чи потрібно створити ключове поле.

6) Для збереження таблиці натиснути кнопку «Зберегти» на панелі інструментів, а потім ввести ім'я таблиці.

При виборі типу даних необхідно враховувати нижче перераховане.

- Які значення повинні відображатися в поле. Наприклад, не можна зберігати текст у поле, що має числовий тип даних.
- Скільки місця необхідно для збереження значень у поле.
- Які операції повинні виконуватися зі значеннями в полі. Наприклад, підсумовувати значення можна в числових полях і в полях, що мають грошовий формат, а значення в текстових полях і полях об'єктів OLE не можна.
- Чи потрібне сортування чи індексування атрибутів. Сортувати й індексувати поля об'єктів OLE неможливо.
- Чи необхідно використання полів в угрупованні записів у запитах чи звітах. Поля об'єктів OLE використовувати для угруповання записів не можна.
- Яким чином повинні бути відсортовані значення в полі. Числа в текстових полях сортуються як рядки цифр (1, 10, 100, 2, 20, 200 і т.д.), а не як числові значення. Для сортування чисел як числових значень необхідно використовувати числові поля чи поля, що мають грошовий формат. Також багато форматів дат неможливо відсортувати належним чином, якщо вони були введені в текстове поле. Для забезпечення сортування використовуйте поле типу «Дата/час».

На рис.2.1 таблиця «Культура» містить два поля:

1) *код культури*; має тип «Текстовий», оскільки може починатися з нуля і немає необхідності проводити математичні обчислення; обмеження довжини складає три символи;
2) *назва культури;* має тип «Текстовий»; максимальна кількість символів складає 255.

Первинним ключем цієї таблиці є поле *код культури*.

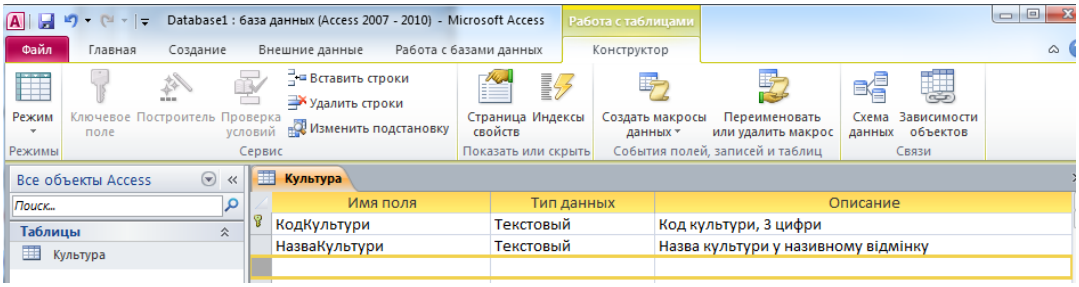


Рис.2.1 Структура таблиці у режимі конструктора Microsoft Access

2.4 Схема бази даних

Access дозволяє встановити ***зв'язок між таблицями***, який визначає тип відношень між полями таблиць. Як правило, зв'язують ключове поле однієї таблиці із відповідним полем іншої таблиці, яке називають полем зовнішнього ключа. Поля, між якими встановлюється зв'язок, можуть мати різні імена, але однаковий тип даних і однакові властивості. Якщо зв'язок між таблицями встановлений, Access буде автоматично вибирати зв'язані дані у звітах, запитах та формах. Приклад схеми бази даних преставлено на рис.2.2.

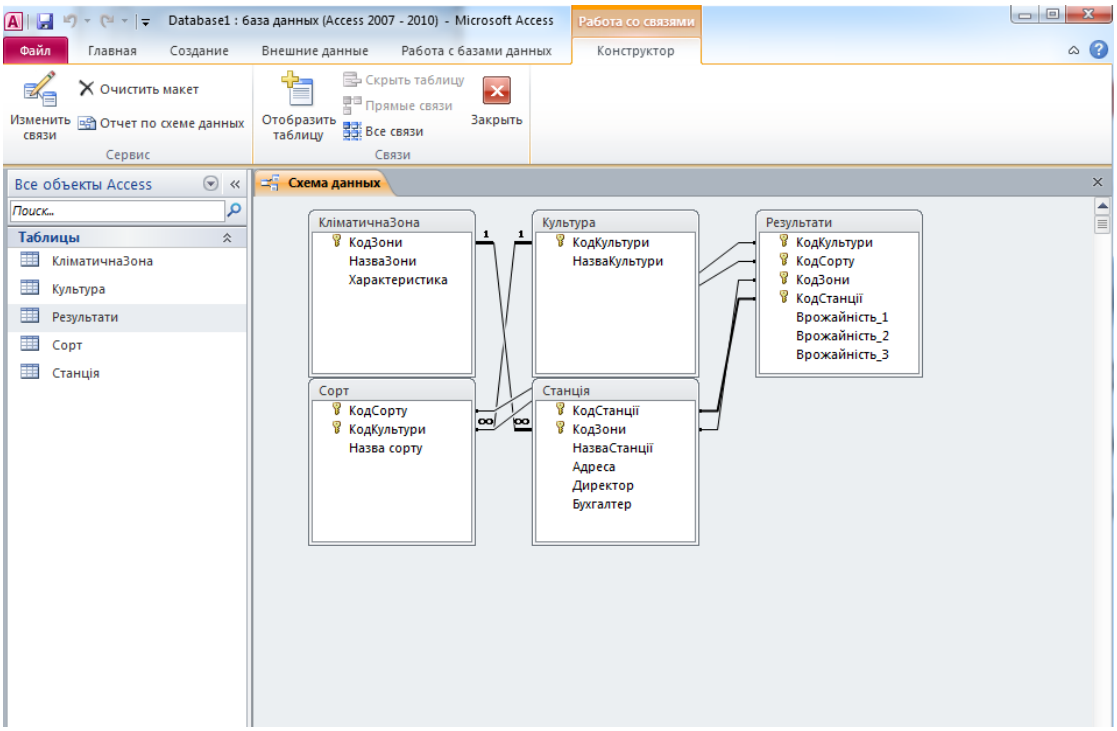


Рис.2.2 Схема бази даних в Microsoft Access

Як видно на рис.2.2, на схемі можна побачити, з яких таблиць складається база даних, структуру кожної із таблиць, включаючи ключові поля або первинні ключі, зв’язки між таблицями та типи цих зв’язків.

Розрізняють три основних типи зв’язків між таблицями (їх також називають відношеннями між таблицями). Для пояснення цих типів будемо вважати, що у нас є дві таблиці – таблиця А і таблиця В.

У відношенні ***«один-до-багатьох»*** кожному запису в таблиці А може відповідати декілька записів у таблиці В, але запис у таблиці В не може мати більш, ніж один відповідний запис у таблиці А. Таке відношення можливе, якщо у таблиці А поле (поля), за яким відбувається зв’язок, є первинним ключем таблиці А, а відповідне поле (поля) таблиці В таким не є. Після визначення такого відношення відповідне поле (поля) таблиці В стає зовнішнім ключем.

У відношенні ***«один-до-одного»*** запис у таблиці А може мати лише один зв’язаний запис у таблиці В, та, навпаки, у таблиці В одному запису відповідає лише один запис у таблиці А. Таке відношення можливе лише тоді, коли в обох таблицях поля, що використовуються для зв’язування, є ключовими.

У відношенні ***«багато-до-багатьох»*** одному запису в таблиці А може відповідати декілька записів в таблиці В, а одному запису в таблиці В – декілька записів в таблиці А. Цей тип зв'язку можливий лише за допомогою третьої таблиці, первинний ключ якої складається з двох полів, які є зовнішніми ключами таблиць А и В. Відношення «багато-до-багатьох» представляє собою два відношення «один-до-багатьох» з третьою таблицею.

Контрольні питання

1. Яку архітектуру має MS Access? Вкажіть недоліки такої архітектури.

2. Які об’єкти входять до складу бази даних MS Access? Який об’єкт є головним?

3. Опишіть основні типи даних MS Access, їх розмір та використання. Для яких типів неможливо визначити ключове поле?

4. Дайте визначення первинного ключа. Яка основна мета його використання?

5. Опишіть процес створення таблиць в Microsoft Access. Які є варіанти цього процесу?

6. Що зображується на схемі даних?

7. Які поля називаються зовнішніми ключами?

8. Поясніть, які типи відношень представлені на рис.2.2. Чи є там відношення «багато-до-багатьох»? Якщо є, то між якими таблицями?

3 ПРОЕКТУВАННЯ ФОРМ

3.1 Призначення форм MS Access

Форми – це інструмент, що використовується для розробки *інтерфейсів користувача* при роботі з базою даних MS Access.

Форми можна використовувати в різних цілях: для введення даних в таблицю, для відкриття інших форм та звітів, для створення діалогового вікна з метою надання користувачеві можливості вибору певних прикладних функцій тощо.

З іншого боку, форма – це об'єкт бази даних, тому зберігається у тому самому файлі, що і таблиці.

Як і інші об'єкти, форму можна розробити як у режимі майстра (рис.3.1), так і у режимі конструктора (рис.3.2). Як правило, спочатку форму розроблюють за допомогою майстра, а у подальшому корегують її у режимі конструктора.

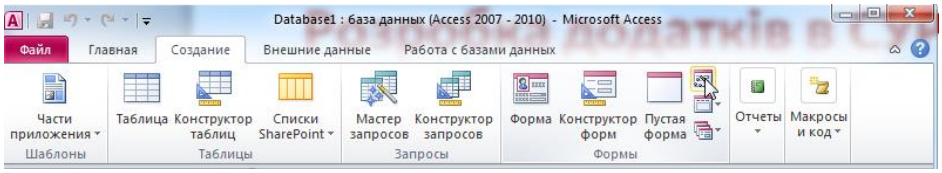


Рис. 3.1 Перехід у режим створення форми за допомогою майстра

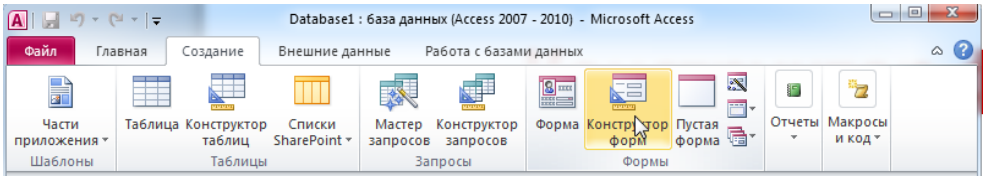


Рис. 3.2 Перехід у режим створення форми за допомогою конструктора

З точки зору інтерфейсу користувача, будь-яка форма – це стандартне вікно Windows. Таке вікно, у свою чергу, складається з стандартних елементів управління, таких, як кнопка, перемикач, прапорець, напис тощо. Оскільки форма – це об'єкт бази даних, є ще додаткові елементи управління, такі, як *поле* та *поле із списком*. Ці

елементи безпосередньо пов’язані з відповідними полями таблиці бази даних.

Елементи управління форм представлено на рис. 3.3.

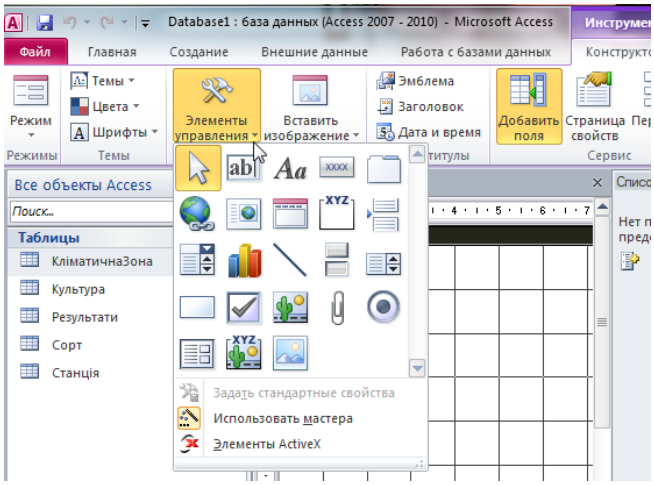


Рис. 3.3 Елементи управління форми

Кожний з елементів управління має власні властивості, за допомогою яких можна управляти елементом з точки зору його зображення на екрані, визначення джерела даних для нього, поведінки при виникненні тої чи іншої події тощо.

3.2 Створення форми за допомогою майстра форм

Поставимо перед собою мету розробити форму для внесення даних у таблицю бази даних. Як було зазначено вище, цей процес краще розпочати у режимі майстра (рис.3.1).

На першому кроці майстер запропонує вибрати таблицю або таблиці та відповідні поля, що будуть відображені на формі. Це представлено на рис. 3.4. Як бачимо, необхідно розробити форму для внесення даних у таблицю «Кліматична зона». На формі будуть представлені поля цієї таблиці (для вибору усіх полів одночасно необхідно натиснути кнопку із зображенням «>>»).

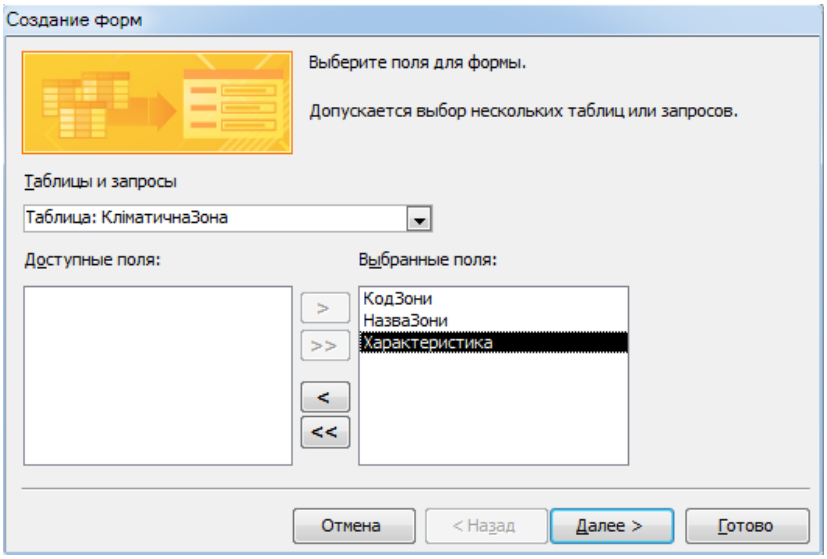


Рис. 3.4 Вибір таблиці та полів, що будуть відображені на формі

На другому кроці майстер запропонує визначитися щодо способу подання елементів управління на формі. Вибір способу подання елементів на формі представлено на рис. 3.5.

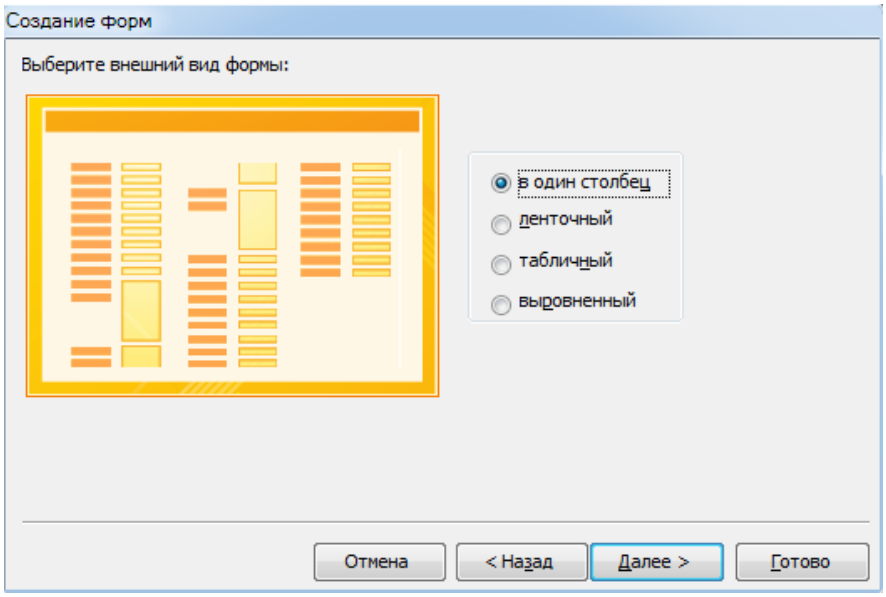


Рис. 3.5 Вибір способу подання елементів на формі

Спосіб подання елементів форми не впливає на функціональні можливості інтерфейсу, а лише покращує (чи погіршує) сприйняття інформації на екрані.

На останньому кроці необхідно визначити ім’я форми. Це представлено на рис. 3.6. Як бачимо, ім’я форми має вказувати на операцію, яку за допомогою цієї форми може виконати користувач.

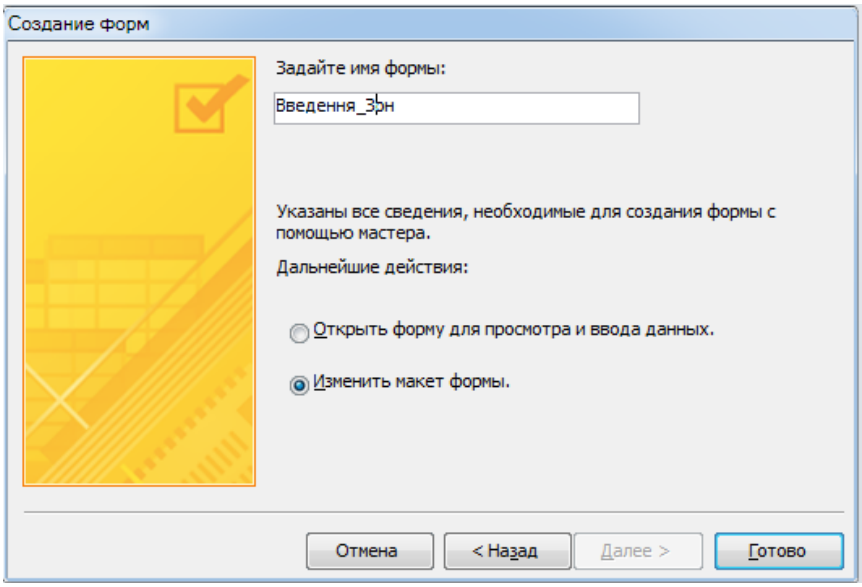


Рис.3.6 Визначення імені форми

На цьому ж кроці, перед натисканням на кнопку «Готово», необхідно визначитися щодо подальших дій зі створеною формою. Як правило, обирається опція «Змінити макет форми» з метою внесення певних корегувань у дизайн створеної майстром форми.

3.3. Робота у режимі конструктора форм

У режимі конструктора форм надається можливість працювати із макетом форми (рис.3.7). Як було зазначено вище, макет форми – це сукупність елементів управління, кожний з який керується своїми властивостями. Сама форма – елемент управління батьківського рівня, для якого також є необхідність змінювати певні особливості у розрізі дизайну, джерела даних, способу роботи з даними тощо. Так, наприклад, за допомогою властивостей форми можна або дозволити, або заборонити внесення нових даних, видалення або корегування записів таблиці.

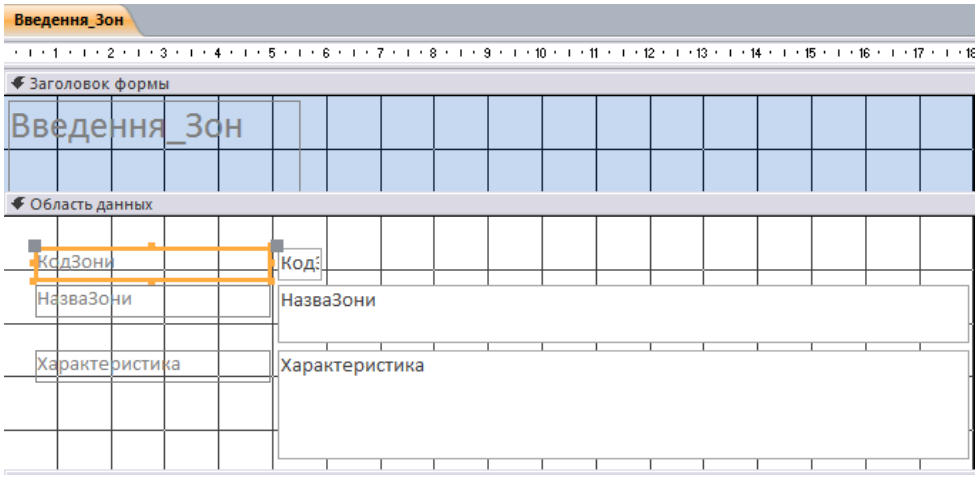


Рис.3.7 Макет форми

Щоб редагувати властивості, необхідно вибрати відповідний елемент управління та відкрити за допомогою правої кнопки миші вікно властивостей як на рис.3.8.

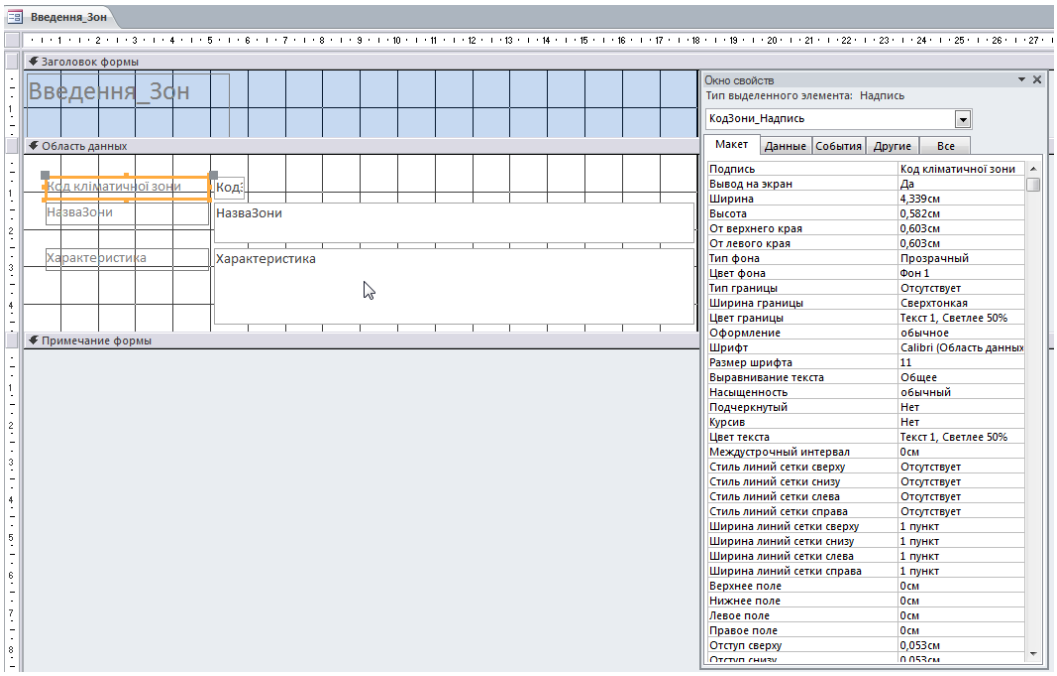


Рис.3.8 Корегування властивостей елементів форми

Як видно з рис.3.8, вікно властивостей поділено вкладками на окремі групи: «Макет», «Дані», «Події», «Інші» та «Усі». В залежності від елемента управління, зміст кожної групи буде відрізнятися. Наприклад, для елемента управління «Напис» відсутні властивості з групи «Дані», елемент управління «Поле» має такі властивості, але їх

значно менше, ніж у елемента управління «Поле із списком». Що стосується властивостей групи «Події», то вони дозволяють реалізовувати певні алгоритми реагування на виникнення тих чи інших подій. Це реалізується за допомогою макрокоманд, макросів, функцій, програмних модулів мовою VBA. Найчастіше властивості цієї групи використовуються для елементу управління «Кнопка» (подія «Натискання кнопки»).

Властивість, що дозволяє реалізувати спливаючу підказку, є у всіх елементів управління. Приклад налаштування спливаючої підказки, яке здійснюється у вікні властивостей елементу управління з вкладкою «Інші», зображено на рис. 3.9.

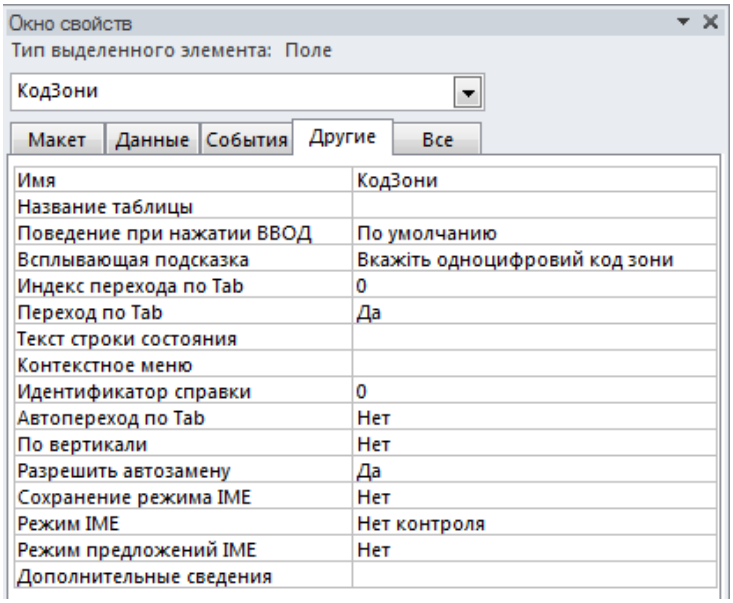


Рис.3.9 Спливаюча підказка

Елемент управління «Кнопка». Майстер форм не дозволяє викласти на форму елемент управління «Кнопка», але це можна зробити у вікні конструктора форм. Для цього в елементах управління необхідно обрати елемент «Кнопка» та здійснити налаштування, наприклад, такі, які представлені на рис. 3.10.

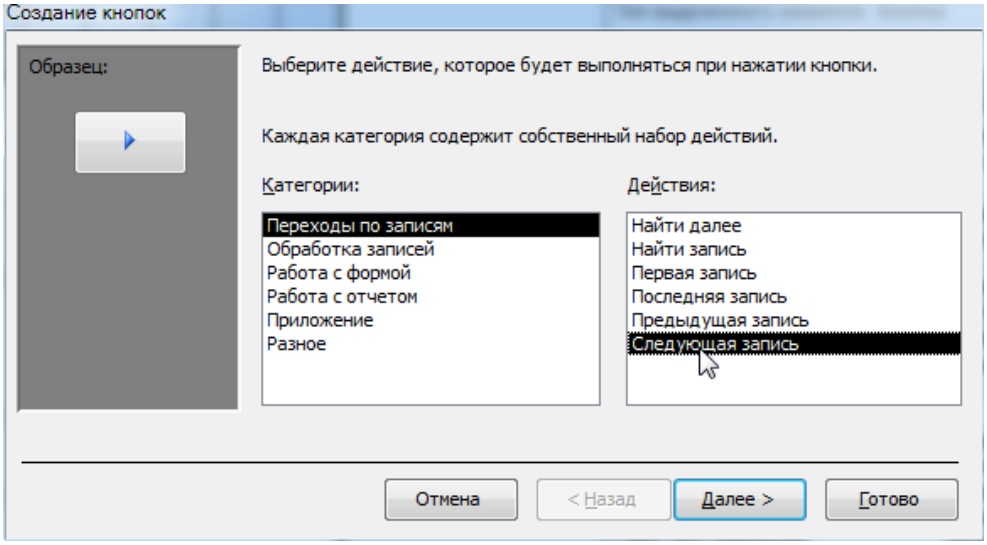


Рис.3.10 Створення кнопок

Подальше оформлення елемента управління «Кнопка» представлено на рис 3.11.

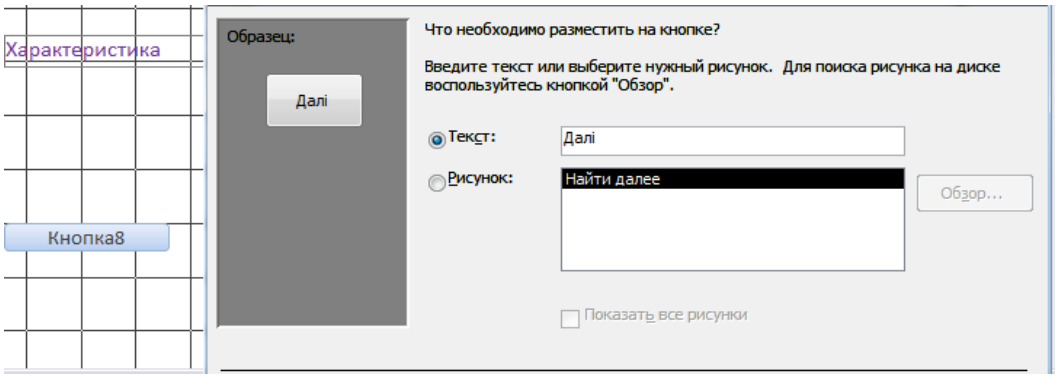


Рис.3.11 Оформлення кнопок

Вигляд створеної форми, так, як її побачить користувач, зображено на рис.3.12.

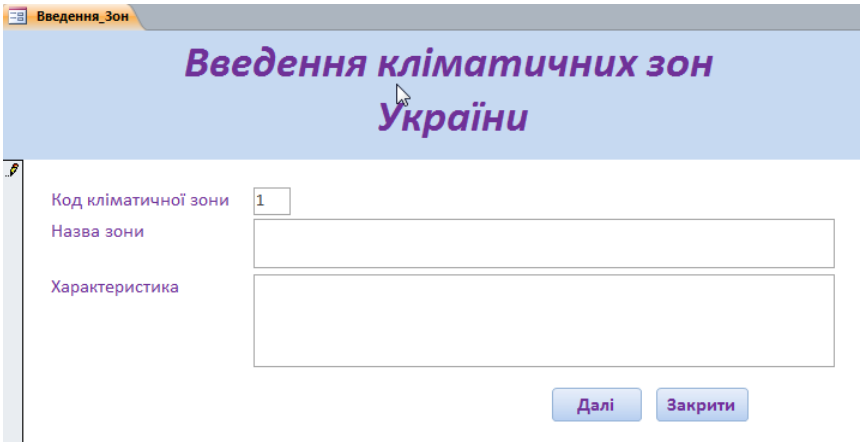


Рис.3.12 Вигляд форми

Поле із списком. До сих пір ми розглядали внесення нових даних шляхом введення відповідних значень у елемент управління «Поле». Але іноді буває необхідним вибрати значення із списку. Такий список може бути сформований або у режимі конструктора (статичний список), або динамічно побудований шляхом вибору значень з таблиць бази даних. У прикладі, що буде розглядатися нижче, інформація щодо кліматичних зон була внесена у таблицю «Кліматична зона». У кожному запису таблиці «Станція» необхідно вказати код кліматичної зони, що дозволить розподілити станції за цими зонами. Такі коди і мають бути вибрані із списку кодів, який формується з таблиці «Кліматична зона». Це реалізується за допомогою елемента управління «Поле із списком». Для побудови елемента управління «Поле із списком» необхідно на елементі «Поле», який майстер виклав на форму, натиснути правою кнопкою миші і змінити його на елемент управління «Поле із списком» (рис.3.13).

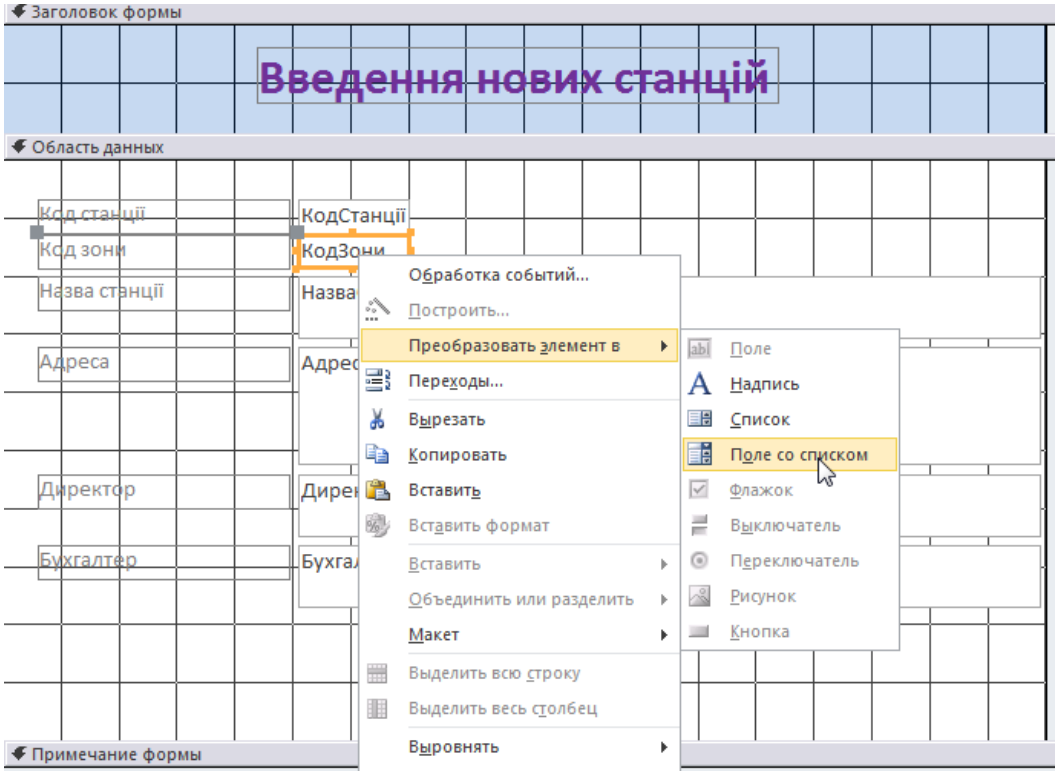


Рис.3.13 Побудова поля із списком

Наступним кроком необхідно обрати джерело формування списку (рис.3.14).

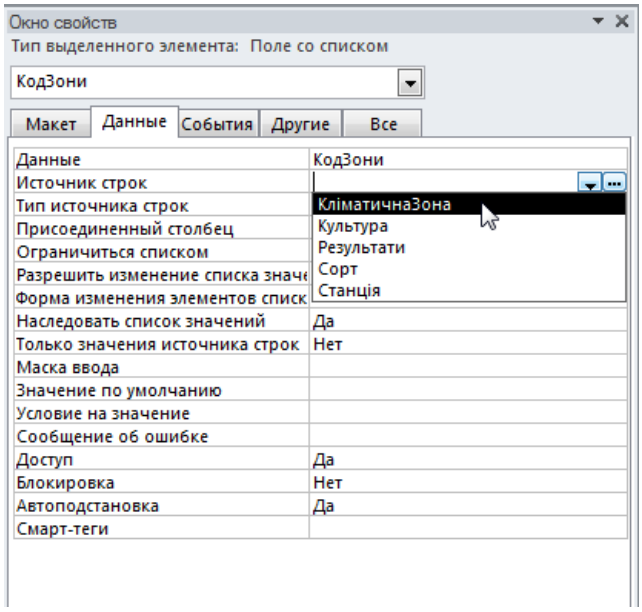


Рис.3.14 Формування списку

Для побудова структури списку необхідно натиснути на три крапки властивості «Джерело рядків» та визначити таблицю або таблиці та поля цієї таблиці, що будуть відображені у списку (рис.3.15).

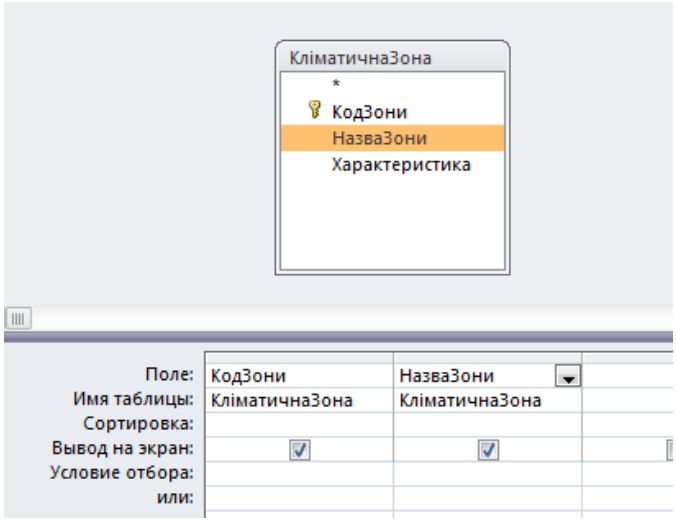


Рис.3.15 Побудова структури списку

Як бачимо, для побудови списку були використані два поля – поле «Код Зони» та поле «Назва Зони». Другий стовпчик слугує підказкою користувачеві, але у таблицю «Станція» не заноситься. Для вказівки того, що заноситься лише перший стовпчик, необхідно скористатися властивістю «Приєднаний стовпчик» групи «Дані», а щоб визначити кількість стовпців у списку, необхідно використати властивості із групи

«Макет». Для макетування списку необхідно змінити кількість стовпців, як це показано на рис.3.16.

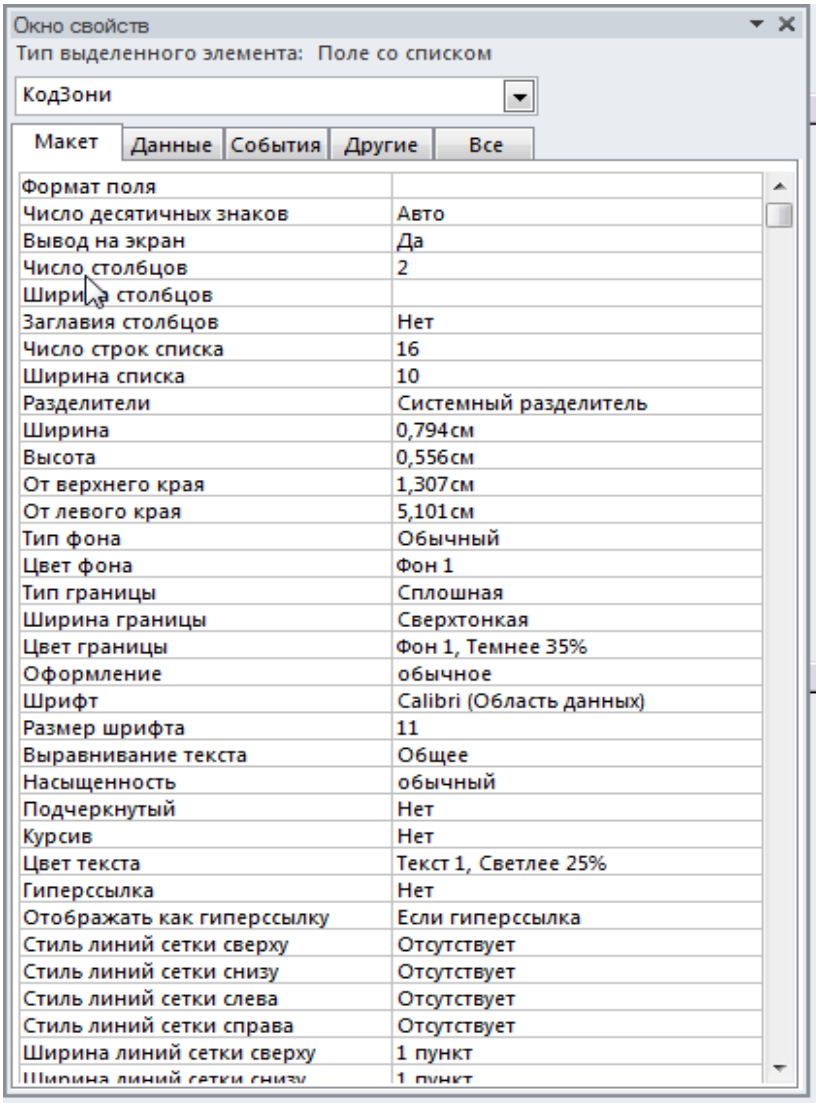


Рис.3.16 Макетування списку

Контрольні питання

1. Поясніть призначення об’єкта бази даних «Форма».

2. Як можна управляти дизайном та функціональністю форми?

3. За допомогою яких режимів можна створити форми в Microsoft Access? У чому полягає різниця між ними?

4. Чи потрібно додатково зберігати дані, які внесені у таблицю за допомогою елемента управління форми «Поле»?

5. Як створити елемент управління «Поле із списком»?

6. За допомогою якої властивості визначається джерело даних для елемента управління «Поле із списком»?

7. Як створити спливаючу підказку?

8. Які властивості є у всіх елементів управління?

9. Як оброблюються властивості з групи «Події»?

10. Як задати значення за замовченням у будь-якому полі?

4 ПРОЕКТУВАННЯ ЗАПИТІВ І ЗВІТІВ

4.1 Призначення запитів MS Access

Термін *запит* у розрізі баз даних позиціонується як механізм управління даними. Дії, які виконуються з ними, такі: внесення, корегування, видалення та пошук або відбір даних. Відповідно запит як механізм управління даними має надавати можливість виконувати усі перераховані дії. Якщо ми розглядаємо задачу відбору даних, запит можна інтерпретувати як віртуальну таблицю, яка динамічно заповнюється під час виконання запиту із реально існуючих таблиць, але ця таблиця знаходиться лише в оперативній пам'яті. Сам же запит зберігається в осередку бази даних.

У термінах MS Access *запит* – це такий самий об'єкт бази даних, як і *таблиця* або *форма*. Запити MS Access використовуються для перегляду, зміни й аналізу даних різними способами. Запити також можна використовувати як джерела записів для форм, звітів і сторінок доступу до даних.

Найбільш розповсюджений тип запитів MS Access — ***запит на вибірку***. Запит на вибірку відбирає дані з однієї чи більше таблиць за заданими умовами, а потім відображає їх у потрібному порядку. Важливо підкреслити, що запит на вибірку не впливає на стан бази даних, а лише дозволяє користувачеві у зручній для нього формі передивлятися дані та аналізувати їх. Для аналізу даних у запиті передбачено використання, так званих, агрегатних функцій, які дозволяють знаходити суму значень, середнє значення, максимальне або мінімальне значення у розрізі стовпчика по всіх рядках таблиці.

4.2 Створення запитів у режимі майстра

Як і попередні об’єкти (таблиці і форми), запити можна створити за допомогою майстра запитів або у режимі конструктору. На рис.4.1 зображено вікно майстра створення запиту.

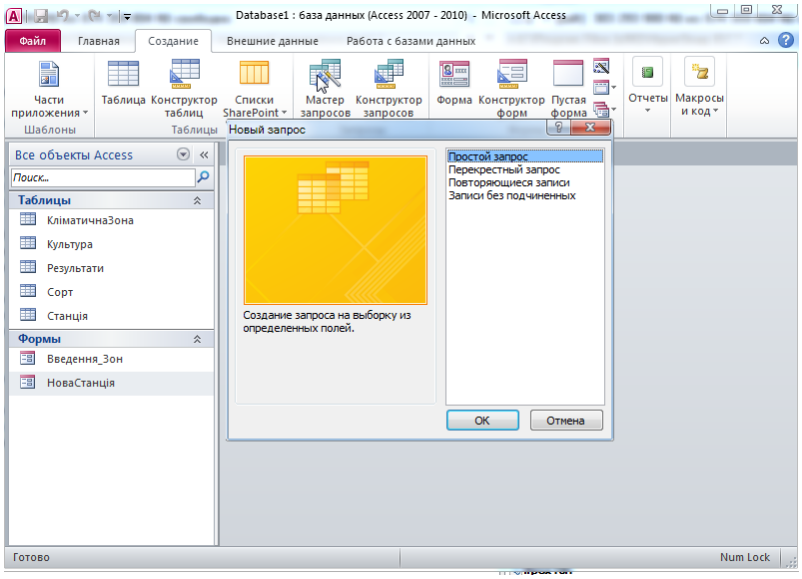


Рис.4.1 Майстер запитів

Як видно з рисунку, обраний «Простий запит».

На першому кроці необхідно визначитися з таблицями та (або) сформованими раніше запитами, які будуть слугувати джерелами даних. Для кожної таблиці або запиту визначаються поля, значення яких необхідно вивести на екран. Створення простих запитів і виконується для вибору полів (рис.4.2).

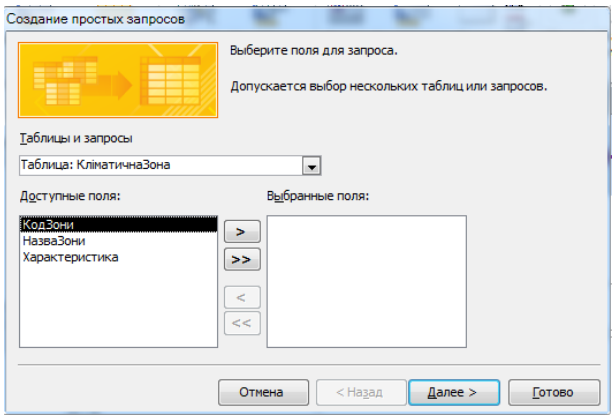


Рис.4.2 Створення простих запитів

На останньому кроці необхідно визначити назву запиту та подальші дії з ним (рис.4.3). Якщо необхідно змінити запит, обирається опція «Змінити макет запиту», і відбувається перехід у режим конструктора.

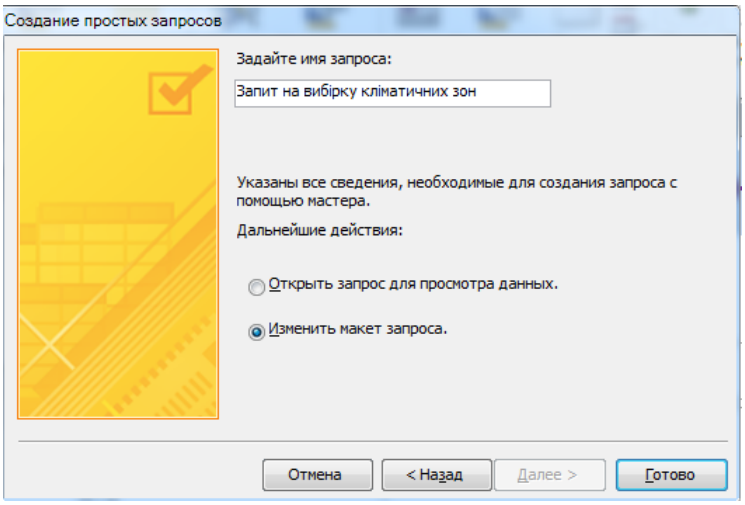


Рис.4.3 Визначення назви запитів

4.3 Створення запитів у режимі конструктора

У режимі конструктора відбувається робота у вигляді заповнення, так званого, бланку запитів (рис.4.4).

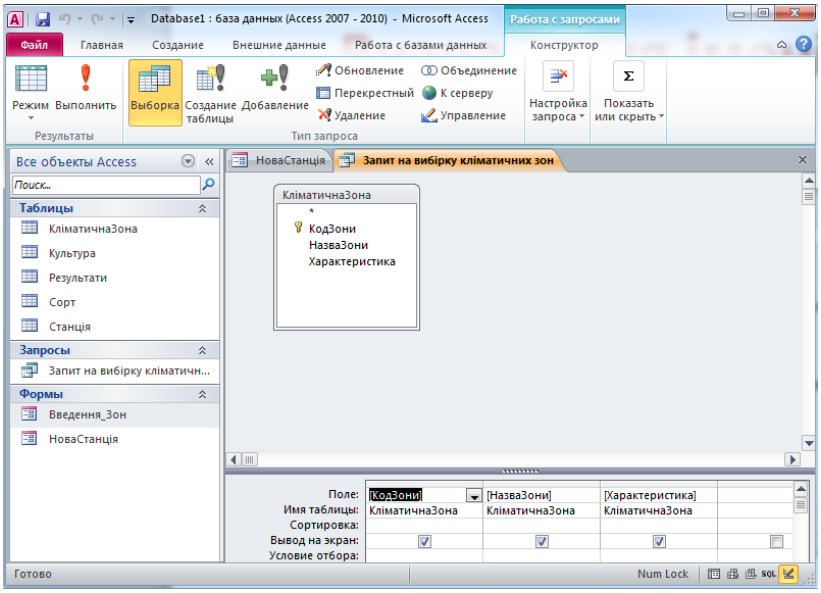


Рис.4.4 Бланк запиту

Бланк запитів організаційно складається з двох частин – верхньої частини (для вибору необхідних для запиту таблиць та інших запитів) і

нижньої частини (для уточнення параметрів виведення на екран та умов відбору у розрізі кожного поля із вибраних таблиць та запитів).

Розробник запитів може за допомогою параметрів виведення на екран визначити умови упорядкування та заборонити чи дозволити виведення значень усередині одного стовпця. За допомогою умов відбору розробник може вказувати значення для стовпця, маючи на увазі виведення лише тих записів, у яких вказаний стовпець має саме це значення. При цьому можна використовувати сталі значення, вирази та діалогові вікна, формуючи тим самим *запит з параметром*.

Після виконання сформованого запиту отримаємо результат у вигляді таблиці, як, наприклад, на рис.4.5.

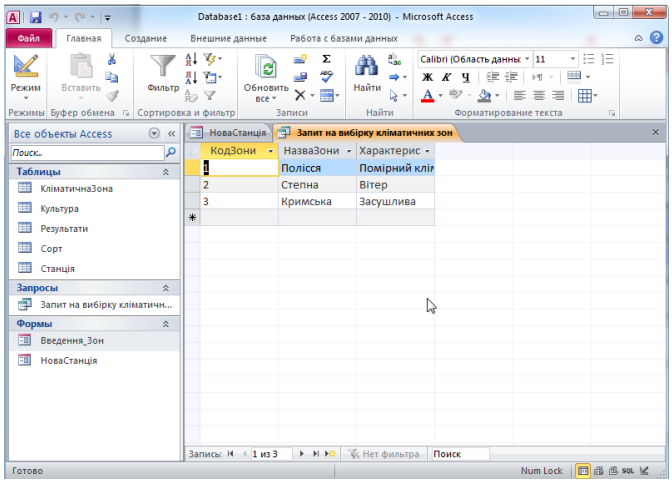


Рис.4.5 Запит на вибірку – перегляд у режимі таблиці

Запит з параметрами та створення діалогових вікон. Як зазначалось раніше, запит, що містить певні умови відбору у розрізі стовпця таблиці, називається запитом з параметром. Сталі значення, які вказуються у нижній частині бланку запитів, потребують кожного разу при зміні цих значень корегування запиту у режимі конструктора. Це вносить певні складнощі у роботу кінцевого користувача. Для реалізації більш гнучкого механізму визначення значень стовпців використовуються діалогові вікна. Їх сформувати дуже просто – у рядку «Умова відбору» в осередку певного поля (нижня частина бланку

запитів) необхідно ввести текст запрошення, обмежений квадратними дужками. Текст запрошення повинний відрізнятися від імені поля, але може включати його.

На рис. 4.6 наведений приклад створення діалогового вікна для визначення значення стовпця «Код зони». В осередку цього поля у квадратних дужках вписаний текст, який при кожному запуску запиту буде з’являтися у заголовку діалогового вікна.

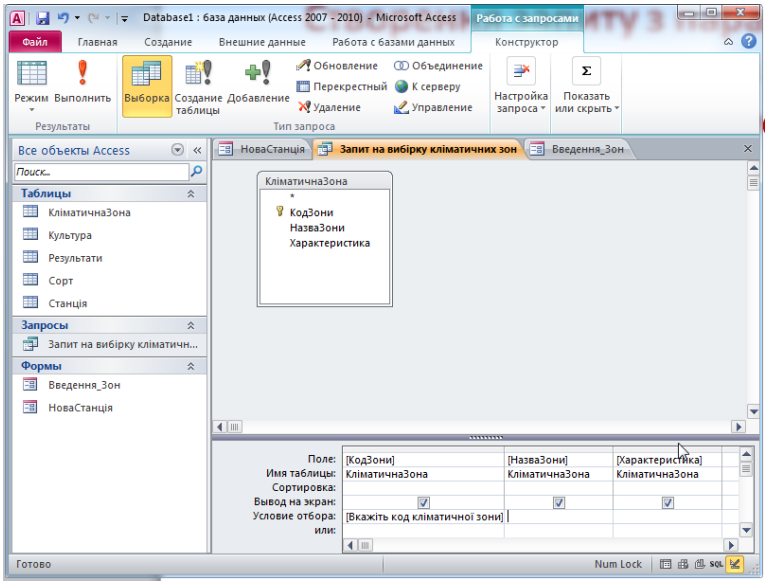


Рис.4.6 Запит з параметром, що запитує введення значення у діалоговому вікні

У результаті вище перерахованих дій, користувач при кожному запуску запиту має можливість вказати необхідний код кліматичної зони. Це показано на рис.4.7.

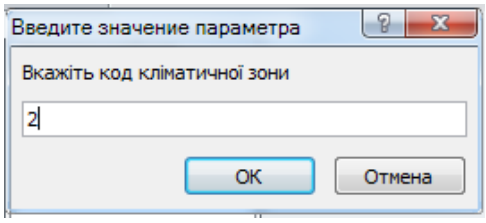


Рис.4.7 Діалогове вікно

У результаті виконаних дій користувач отримує таблицю як на рис.4.8.

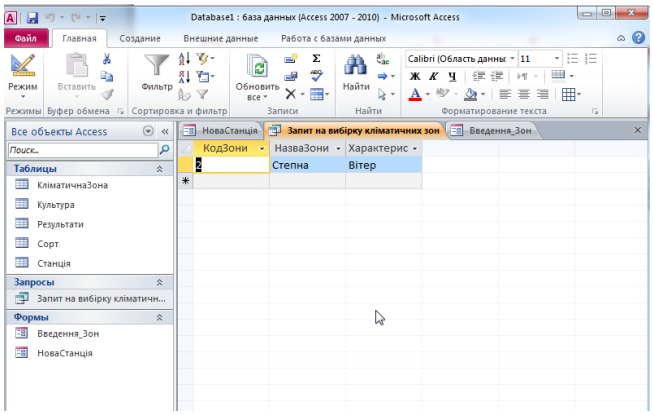


Рис.4.8 Результат відбору

Використання конструкції «Between». Якщо необхідно задати умову із діапазоном значень, зручно використовувати конструкцію «Between». Наприклад, для поля, що виводить дати, можна використовувати цю конструкцію для визначення діапазону дат. В осередку «Умова відбору» такого поля необхідно вказати: «Between [Уведіть початкову дату:] And [Уведіть кінцеву дату:]».

Використання конструкції «Like». Щоб запросити у користувача один чи кілька символів для пошуку записів, що починаються з цих символів чи містять їх, доцільно створити запит з параметрами, що використовує конструкцію «Like» і знаки (&,*,?). Наприклад, вираз:

LIKE [Уведіть перший символ для пошуку:] & ""*

виконує пошук слів, що починаються з зазначеного символу. А вираз:

LIKE "" & [Уведіть будь-який символ для пошуку:] & "*"*

виконує пошук слів, що містять зазначений символ. Вираз:

LIKE "?" & [Уведіть будь-який символ для пошуку:] & ""*

виконує пошук слів, у яких на друій позиції знаходиться вказаний символ.

На рис.9 показано, як задати вибірку по назвах кліматичних зон, що починаються з літери «П».

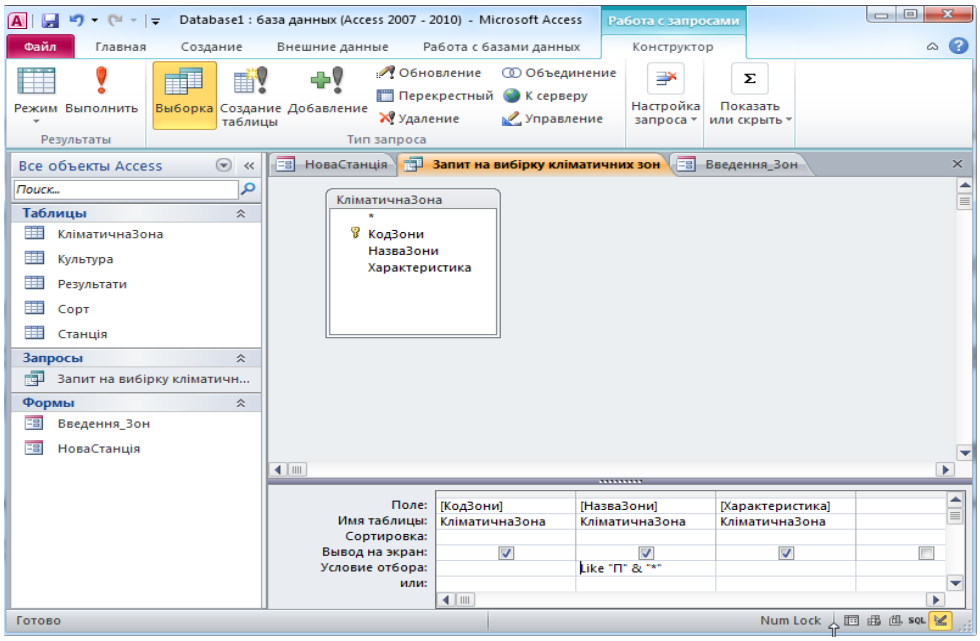


Рис.4.9 Вибірка з використанням оператора LIKE

Перехресні запити і їхнє використання. У перехресному запиті відображаються результати статистичних розрахунків (суми, кількість записів і середні значення), виконаних за значеннями з одного полю таблиці. Ці результати групуються за двома наборами даних, один із яких розташований у лівому стовпці таблиці, а другий — у верхньому рядку.

Дані перехресної таблиці можна вивести і без створення окремого запиту в базі даних — або за допомогою майстра зведених таблиць у формі, або створивши зведений список на сторінці доступу до даних. При використанні зведеної форми чи зведеного списку можна змінювати заголовки рядків і стовпців у міру необхідності, щоб аналізувати дані з різних сторін.

Запити на зміну і їхнє використання. Запитом на зміну називають запит, що за одну операцію вносить зміни в кілька записів. Існує чотири типи запитів на зміну: на видалення, на відновлення і додавання записів, а також на створення таблиці.

Перегляд запиту у режимі SQL. Насправді, усі запити у реляційній базі даних мають бути записані на мові SQL (див. розділ 5). Як ми побачили, MS Access дозволяє побудувати запити, не використовуючи цю мову, у режимах майстра або конструктора. Та, насправді, MS Access, автоматично генерує код запиту на мові SQL відповідно діям розробника запитів у режимах майстра та конструктора. Щоб переглянути запит у режимі SQL, необхідно змінити подання запиту як на рис.4.10.

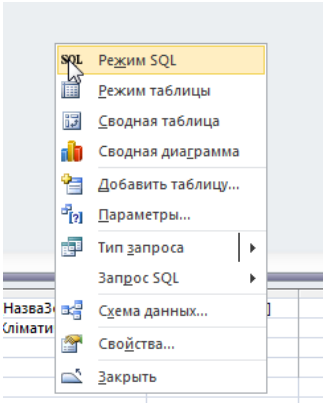


Рис.4.10 Перегляд запиту в режимі SQL

Запит, який показаний на рисунках 4.4-4.5, у режимі SQL буде виглядати так:

```
SELECT КліматичнаЗона.[КодЗони],
КліматичнаЗона.[НазваЗони], КліматичнаЗона.[Характеристика]
FROM КліматичнаЗона
```

Запит, який показаний на рисунках 4.6-4.8, у режимі SQL буде виглядати так:

```
SELECT КліматичнаЗона.[КодЗони],
КліматичнаЗона.[НазваЗони], КліматичнаЗона.[Характеристика]
FROM КліматичнаЗона
WHERE КліматичнаЗона.[КодЗони]=[Вкажіть код кліматичної
зони]
```

Запит з використанням конструкції «LIKE» в режимі SQL буде виглядати так:

```
SELECT КліматичнаЗона.[КодЗони],
КліматичнаЗона.[НазваЗони], КліматичнаЗона.[Характеристика]
FROM КліматичнаЗона
WHERE (((КліматичнаЗона.[НазваЗони]) Like "П" & "*"))
```

Подання запиту у режимі SQL дозволяє не лише переглядати його, а і змінювати код запиту. Таким чином, MS Access дозволяє зробити процес формування запитів більш гнучким і ефективним.

4.3 Створення звітів

Звіт являє собою ефективний засіб представлення даних в друкованому вигляді. Маючи можливість керувати розміром та зовнішнім виглядом всіх елементів звіту, користувач може відображувати інформацію у бажаному вигляді.

Більша частина інформації у звіті надається із таблиці бази даних, запиту або інструкції SQL, які є джерелом даних для звіту. Інша інформація звіту зберігається у його структурі.

Для створення зв'язку між звітом та його джерельними даними використовуються елементи управління (поля, написи, декоративні лінії для оформлення звіту тощо).

Режими створення звітів представлені на рис.4.11.

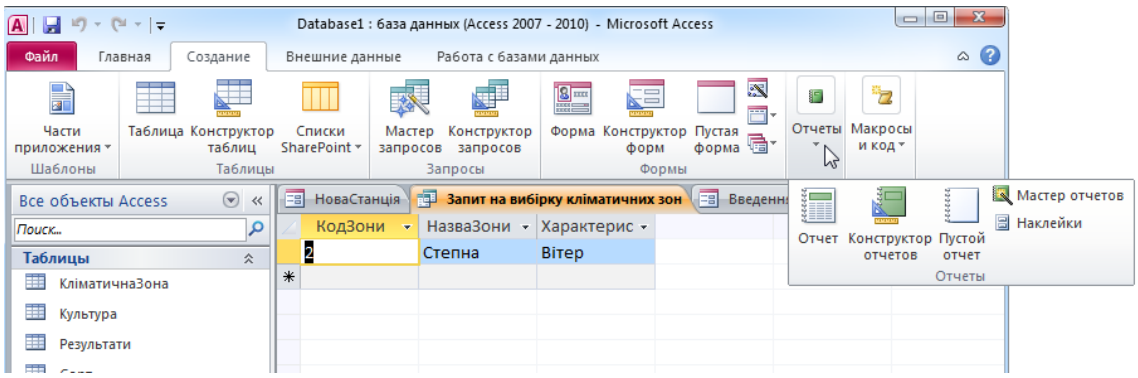


Рис.4.11. Режими створення звітів

Для побудови звіту необхідно у вікні створення звітів обрати запит та поля, які будуть відображені у звіті (рис.4.12).

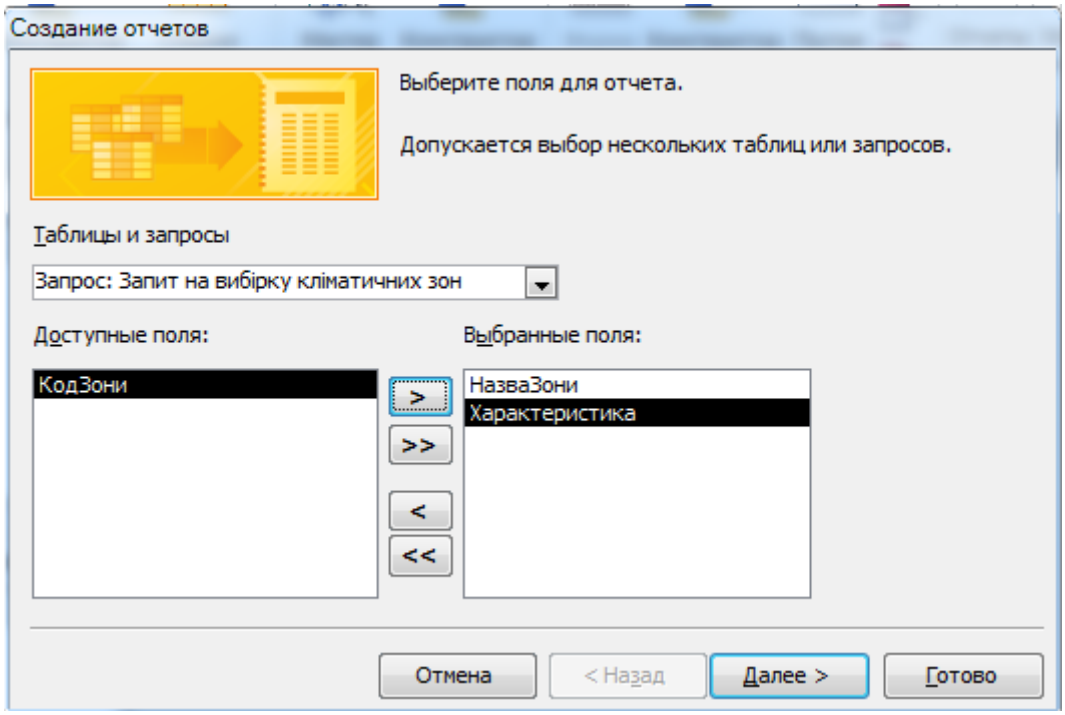


Рис.4.12. Вибір запиту для побудови звіту

Наступним кроком необхідно визначити спосіб групування у звіті (рис.4.13).

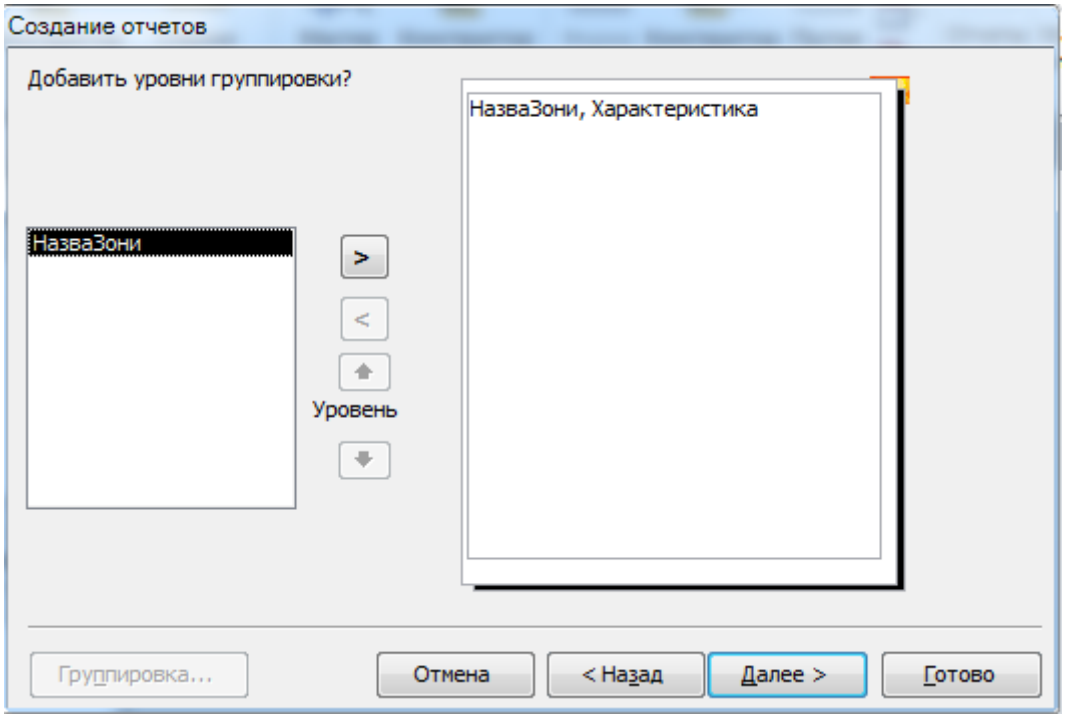


Рис.4.13 Визначення способів групування у звіті

Визначення порядку сортування записів представлено на рис.4.14.

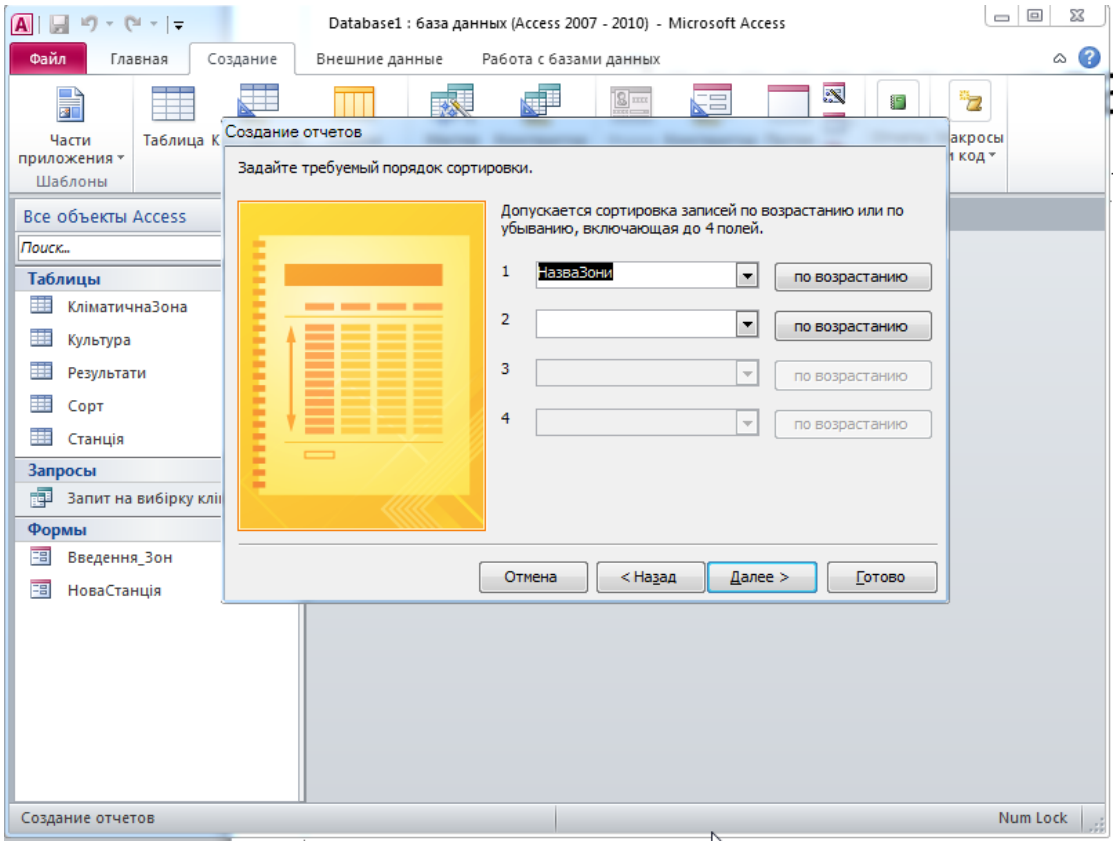


Рис.4.14. Визначення порядку сортування записів

Остаточний вигляд звіту представлено на рис.4.15.

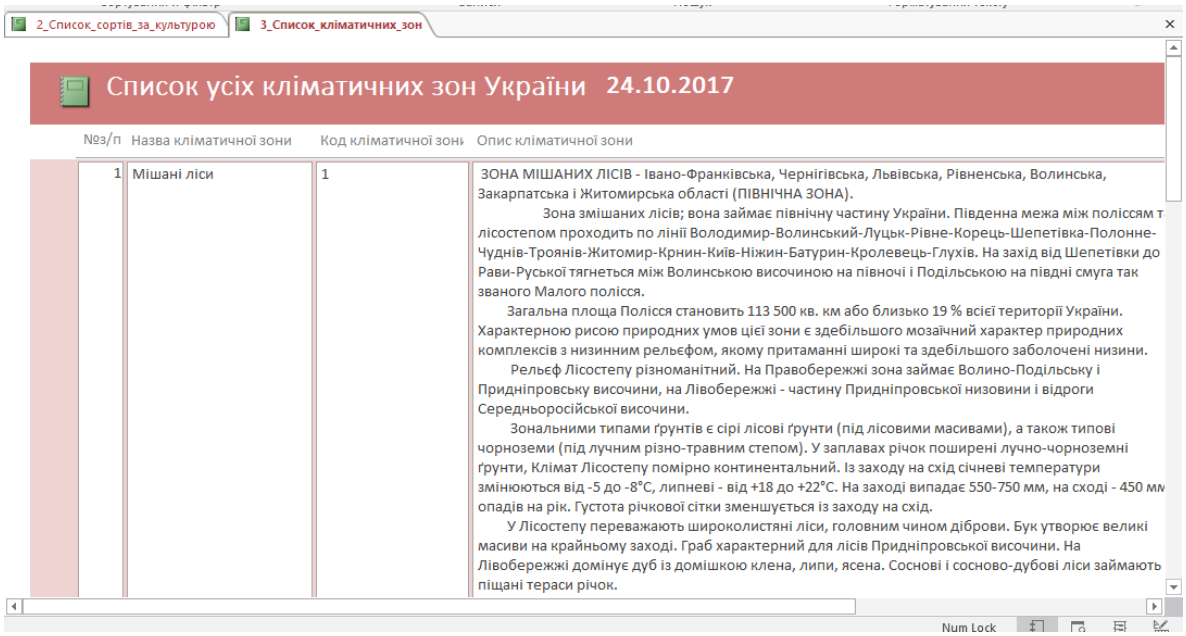


Рис.4.15 Звіт

Контрольні питання

1. Які основні об’єкти складають базу даних MS Access?

2. Для чого використовуються запити у реляційній базі даних?

3. Які типи запитів використовуються у реляційній базі даних?

4. Що означає поняття «запит на вибірку»?

5. У яких режимах можна формувати запит у середовищі MS Access? Чим відрізняються ці режими?

6. Чим відрізняються простий запит та запит з параметром?

7. Як побудувати діалогове вікно для запиту?

8. Побудуйте запит із використанням конструкції «Between» для визначення діапазону числових значень.

9. Для чого призначена конструкція LIKE? Побудуйте запит на вибірку записів з будь-яким значенням певного текстового поля.

10. Поясніть використання звітів.

11. У яких режимах можна створювати звіти?

12. Що дозволяє реалізувати процес групування у звітах?

13. Як змінювати вигляд звіту?

14. Покажіть, як задавати номер за порядком у списках, що представлені у звіті.

15. Які об’єкти можуть слугувати джерелом даних для звітів? Чи можуть бути серед них інші звіти?

5 Мова SQL

5.1 Основні засади

Історія мови SQL починається з 1970 року, коли в дослідницькій лабораторії IBM в штаті Каліфорнія була розроблена її перша версія. Назва мови є абревіатурою від слів Structured Query Language (структурована мова запитів) і іноді це вимовляють як "sequel" (первинна назва). Спочатку ця мова була реалізована в реляційній СУБД DB2 фірмою IBM. SQL є не процедурною мовою на відміну від процедурних мов третього покоління (third-generation languages -3GL), таких як COBOL і C, які з'явилися в цей же час. Відзначимо, що не процедурність мови означає, що в ній формулюється те, що необхідно отримати, а не як це зробити.

Специфічною особливістю реляційних систем управління базами даних (РСУБД) є те, що вони надають мову маніпулювання базами даних, яка орієнтується на множини (set-oriented database language). Для переважної більшості сучасних СУБД такою мовою є SQL.

Дві організації - Американський Національний Інститут Стандартів - AHCI (American National Standards Institute – ANSI) і Міжнародна Організація Стандартів - ICO (International Standards Organization – ISO) займаються описом і підтримкою стандартів цією мови. Не дивлячись на те, що всі сучасні СУБД підтримують той або інший стандарт, проте у кожному конкретному випадку допускаються відхилення, які завжди специфікуються в документації. Більш того, більшість систем пропонують розширення цієї мови, які допускають можливість її використання в інших процедурних мовах.

Не дивлячись на те, що мова називається мовою запитів, вона надає наступні можливості:

- визначати, перевизначити і видаляти таблиці бази даних,

- вставляти, змінювати і видаляти рядки в таблицях,
- проводити пошук даних в багатьох таблицях і упорядковувати результати пошуку,
- описувати процедури підтримки цілісності,
- визначати і змінювати інформацію про захист даних.

Суть мови розкриватиметься в прикладах її використання. За основу викладу прийнятий стандарт ANSI-92. Всі запити формулюватимуться по відношенню до схеми бази даних, яка представлена на рис.2. База даних складається з 7-ми таблиць та містить інформацію про факультет (**FACULTY**), кафедру (**DEPARTMENT**), предмети (**SUBJECT**), викладачів (**TEACHER**), групи (**GROUP**), аудиторії (**ROOM**) та розклад (**LECTURE**). Первинні та зовнішні ключі у кожній із таблиць позначені знаком **#**.

FACULTY (#F, Name, Dean, Building, Fund)

SUBJECT (#S, Name)

DEPARTMENT (#D, OF, Name, Head, Building, Fund)

ROOM (#R, Num, Building, Seats)

GROUP (#G, #D, Course, Num, Quantity, #Curator)

TEACHER (#T, #D, Name, Post, Tel)

LECTURE (#T, #G, #S, #R, Type, Day, Week)

Рис.5.1 Схема учбової бази даних

5.2 Засоби вибірки даних

5.2.1 Основні конструкції мови запитів. Основна конструкція мови запитів складається з фраз **SELECT** і **FROM**. Фраза FROM указує, з якої таблиці слід вибрати дані, а фраза SELECT – які саме атрибути (стовпці) з вказаної таблиці повинні бути виведені. Так, наприклад, запит

```
SELECT Name FROM FACULTY
```

приводить до виведення назв факультетів. Ці дві фрази обов'язково

повинні бути присутніми у будь-якому запиті.

Виведення окремих стовпців. У фразі SELECT можна приводити список імен полів. При цьому передбачається, що результат впорядкований по стовпцях згідно порядку назв у цій фразі:

*SELECT Num, Course, Quantity FROM GROUP***.**

Виведення всіх стовпців. Якщо необхідно вивести всі поля таблиці, то у фразі SELECT використовується символ «*»:

*SELECT * FROM DEPARTMENT***.**

Рядки, що не повторюються. Хоча в таблицях не повинно бути рядків, що повторюються, проте SQL за замовчанням припускає, що якщо в запиті указується окремий стовпець (або група стовпців) і є значення, що повторюються, то всі вони виводяться. Щоб отримати в результаті виконання запиту унікальні (що не повторюються) значення, слід використовувати ключове слово **DISTINCT**. Наприклад, щоб отримати список всіх типів лекцій, що викладаються у вузі, слідує написати:

SELECT DISTINCT Type
*FROM LECTURE***.**

Інакше ми б отримали список з декількох сотень рядків (розмір списку був би рівним всім лекціям у вузі). Звернемо увагу на те, що весь запит може бути записаний в одному рядку.

Зміна імен стовпців. Фраза SELECT надає можливість перевизначити імена стовпців результуючої таблиці. Для цього необхідно слід за ім'ям стовпця початкової таблиці вказати нове ім'я результуючої таблиці. Наприклад, у наступному запиті змінюються імена обох стовпців:

SELECT Name Faculty_name, Dean Faculty_dean
FROM FACULTY**.**

Завдання умови вибірки. Для завдання умови вибірки

використовується ключове слово **WHERE**. У ньому специфікується, якій умові повинні задовольняти вихідні дані. Алгоритм роботи описується таким чином:

- вибирається черговий рядок з таблиці,
- на ній перевіряється вказана умова,
- якщо рядок задовольняє умові, то виводяться значення тих стовпців, які вказані у ключовому слові SELECT.

Наприклад, приведений нижче запит приводить до виведення списку всіх професорів вузу:

```
SELECT  Name FROM   TEACHER
WHERE  Post = 'професор'.
```

5.2.2. Вирази, умови і оператори. Визначення виразу доволі просте: ***вираз*** повертає значення. Типи виразів досить різноманітні, включаючи різні типи даних: рядки, числа, логічні значення. Конструкція SELECT FROM WHERE також є виразом, не говорячи вже про просте входження імені стовпця у слові SELECT, яке повертає значення стовпця. Конструкція «Post = 'професор'» також є виразом, оскільки в результаті отримуємо правдиве значення.

Умова – це вираз, який повертає правдиве значення **TRUE** або **FALSE**. Умовні вирази обов'язково використовуються за словом WHERE, а також можуть використовуватись і в інших фразах, наприклад, SELECT.

Оператори – це конструкції, які використовуються у виразах для вказівки деяких операцій над ними. Розрізняють такі типи операторів: арифметичні, над рядками, логічні, порівняння, теоретико-множинні. У конкретних системах списки цих операторів можуть розширюватись.

- Арифметичні оператори – стандартні оператори +, - *, /.
- Оператори над рядками – вкажемо тільки на один – зчеплення або конкатенація рядків – ||.

- Теоретико-множинні оператори – це *UNION*, *INTERSECT*, *MINUS*.
- Логічні оператори – це *AND*, *OR*, *NOT*.
- Оператори порівняння. Їх список приведений нижче в таблиці 2.

Таблиця 2

Оператори порівняння мови SQL

Оператор	**Призначення**	**Приклад**
=	Перевірка на рівність	SELECT * FROM emp WHERE sal = 1500
!=,^=,<>	Перевірка на нерівність	SELECT * FROM emp WHERE sal != 1500
> <	Перевірка на більше або менше	SELECT * FROM emp WHERE sal >500 SELECT * FROM emp WHERE sal <500
>= <=	Перевірка на більше або рівно і менше або рівно	SELECT * FROM emp WHERE sal >=500 SELECT * FROM emp WHERE sal <=500
IN	Перевірка на входження елементу в множину	SELECT * FROM emp WHERE job [IN ('CLERK', 'ANALYST ') SELECT * FROM emp WHERE sal IN (SELECT sal FROM emp WHERE deptno = 30
NOT IN	Еквівалентний "! = ALL". Перевірка на те, чи елемент не входить в множину.	SELECT * FROM emp WHERE sal NOT IN (SELECT sal FROM emp WHERE deptno = 30) SELECT * FROM emp WHERE job NOT IN ('CLERK', 'ANALYST')

Оператор	Призначення	Приклад
ANY SOME	Слідує за одним з предикатів =, !=, >, <,= <=. Перевіряє чи виконується цей предикат на хоча б одному значенні множини, заданої в правій частині, по відношенню до елементу, який заданий в лівій частині	SELECT * FROM emp WHERE sal = ANY (SELECT sal FROM emp WHERE deptno = 30) ;
ALL	Слідує за одним з предикатів =, !=, >, <, <=, >=. Перевіряє, чи виконується цей предикат на всіх значеннях множини, заданої в правій частині, по відношенню до елементу, який заданий в лівій частині	SELECT * FROM emp WHERE sal >= ALL (1400, 3000);

Оператор	Призначення	Приклад
EXISTS	Результат виконання TRUE, якщо підзапит, до якого оператор засто-совується, містить хоч б один рядок	SELECT ename, deptno FROM dept WHERE EXISTS (SELECT * FROM emp WHERE dept.deptno = emp.deptno);
IS [NOT] NULL	Перевіряє на значення NULL стовпця	SELECT ename, deptno FROM emp WHERE comm IS NULL;

Розглянемо ряд прикладів використання цих операторів.

1. Отримати список номерів груп, в яких кількість студентів знаходиться в межах між 40 і 50:

SELECT Num, Course, Quantity FROM GROUP
WHERE Quantity >= 40 AND Quantity <=50.

2. Показати імена тільки професорів і асистентів:

SELECT Name, Post FROM TEACHER
WHERE Post IN ('професор', 'асистент').

5.2.3. Вибірка з декількох таблиць. Одним з наймогутніших засобів SQL є можливість вибирати дані з багатьох таблиць. Це досягається перерахуванням імен необхідних таблиць після ключового слова FROM. Якщо при цьому у ключовому слові WHERE не вказується умова їх з'єднання, то проводиться декартове перетворення всіх таблиць після ключового слово FROM. Як правило, таблиці поєднуються за деякою умовою. Наприклад, для отримання списку назв факультетів і назв відповідних їм кафедр, необхідно записати:

SELECT FACULTY.Name, DEPARTMENT.Name
FROM FACULTY, DEPARTMENT
WHERE FACULTY.#F= DEPARTMENT.#F.

Уточнення імен стовпців. Звернемо увагу, що у ключових словах

SELECT і WHERE імена стовпців уточнюються іменами таблиць. Якщо є стовпець, який має одне і те ж ім'я в об'єднувальних таблицях, то посилатись на той або інший стовпець в запиті слід саме так – уточнювати їх іменами таблиць.

Аліас імені таблиці. У зв'язку з тим, що таблиці можуть мати довгі назви, мова надає можливість пов'язувати з кожною таблицею деякий короткий аліас, і надалі посилатися на таблицю за ним. Зіставлення таблиці з аліасом визначається за допомогою ключового слова FROM. Наприклад, попередній запит можна записати:

```
SELECT f1.Name, f1.#F, f2.#F
FROM FACULTY f1, FACULTY f2
WHERE f1.Name=f2.Name AND f1.#F!=f2.#F.
```

Очевидно, що ключове слово WHERE може використовуватися одночасно як для вказівки способу з'єднання таблиць, так і для вказівки умови відбору рядків в результуючу таблицю. Наприклад, запит:

```
SELECT DEPARTMENT.Name FROM DEPARTMENT, TEACHER
WHERE DEPARTMENT.#D = TEACHER.#D AND
TEACHER.Name = 'Іванов'
```

приводить до виведення назви кафедри, на якій працює викладач Іванов. (Відмітимо, що пошук викладачів ведеться по всьому вузу, тому по суті відшукуються всі кафедри вузу, на яких працюють викладачі за прізвищем Іванов). Зверніть увагу, що умова пошуку задається на одній таблиці, а результат видається з іншої.

Обчислювані стовпці. У ключовому слові SELECT може бути сформований новий стовпець, значення якого обчислюється з інших стовпців таблиць, що поєднуються. Запит, що приводиться далі, виводить на екран зведення про всі кафедри факультету інформатики з їх фондами фінансування і відсотком, який цей фонд складає по відношенню до фонду факультету:

```
SELECT  d.Name,  d.Fund,  (d.Fund / f.Fund) * 100
FROM      FACULTY f,  DEPARTMENT d
WHERE  f.#F  =  d.#F.
```

Екві-з'єднання. Якщо таблиці з'єднуються за рівністю значень однієї або декількох пар стовпців, причому з кожної таблиці вибираються всі стовпці, то таке з'єднання виражає операцію *екві-з'єднання* реляційної алгебри, наприклад:

```
SELECT  f.*,  d.*FROM  FACULTY f,  DEPARTMENT d
WHERE  f.#F  =  d.#F.
```

Природне з'єднання. Операція реляційної алгебри природного з'єднання виражається з'єднанням двох або декількох таблиць за предикатом рівності з подальшим видаленням стовпців, що повторюються, шляхом явного перерахунку всіх необхідних стовпців у ключовому слові SELECT. Наприклад:

```
SELECT  f.#F,  f.Name, f.Dean, f.Building, f.Fund, d.#D,  d.Name, d.Head,
d.Building, d.Fund FROM      FACULTY f,  DEPARTMENT d
WHERE  f.#F  =  d.#F.
```

Звернемо увагу, що хоча стовпці Name, Building, Fund зустрічаються в обох таблицях, проте вони не розглядаються як ті, що повторюються, оскільки несуть різне семантичне навантаження. Стовпцем, що повторюється, є той єдиний, за яким проводиться з'єднання – стовпець #f, що виконує роль зовнішнього ключа в таблиці DEPARTMENT.

Тета-з'єднання. Це з'єднання за будь-якою іншою умовою, що відрізняється від рівності. В цьому випадку з'єднання проводиться не за первинним або зовнішнім ключем, а за деякими іншими стовпцями і має більш глибоку семантику.

З'єднання таблиці зі своєю копією. В деяких випадках необхідно з'єднати таблицю з собою. В такому випадку після ключового слова

FROM записується двічі ім'я таблиці з різними аліасами, щоб можна було до кожної з них посилатися самостійно. Розглянемо такий приклад: нехай нам необхідно перевірити, чи є в таблиці FACULTY такі пари рядків, в яких імена факультетів співпадають, а їх ключі #F різні. Для цього слід записати:

SELECT f1.Name, f1.#F, f2.#F FROM FACULTY f1, FACULTY f2 WHERE f1.Name=f2.Name AND f1.#F!=f2.#F.

З’єднувати можна і по трьом, чотирьом і т.д. таблицям. Мова не обмежує кількість таблиць, що з’єднується. Але при цьому слід пам'ятати, що вкрай важливо проводити саме з'єднання за деякою умовою, а не декартовий добуток таблиць. Майте на увазі, що якщо провести декартовий добуток п'яти таблиць, кожна з яких містить всього по 20 рядків, то результуюча таблиця матиме досить значний розмір – 3,200,000 рядків. Декартовий добуток двох таблиць з десятками тисяч рядків кожна може обраховуватися годинами. А об’єм результуючої таблиці може обчислюватися десятками гігабайт.

Приведемо декілька запитів, які використовувалися при описі реляційної алгебри, приводячи тексти запитів на мові SQL.

Запит 1. Вивести на екран список всіх кафедр факультету ІТ:

SELECT DEPARTMENT.Name FROM FACULTY, DEPARTMENT WHERE FACULTY.#F = DEPARTMENT.#F AND FACULTY.Name = 'IT'.

Запит 2. Вивести на екран список всіх викладачів з їхніми телефонами кафедри технологій програмування (ТП):

SELECT TEACHER.Name, Tel FROM DEPARTMENT, TEACHER WHERE DEPARTMENT.#D = TEACHER.#D AND DEPARTMENT.Name = 'ТП'.

Запит 3. Вивести на екран список номерів усіх груп першого курсу кафедри ТП:

SELECT Num FROM GROUP, DEPARTMENT
WHERE GROUP.#D = DEPARTMENT.#D AND
Name = 'ТП' AND COURSE = 1.

Запит 4. Вивести на екран список номерів всіх груп першого курсу кафедри ТП разом з кураторами цих груп:

SELECT Num, TEACHER.Name FROM GROUP, DEPARTMENT,
TEACHER WHERE GROUP.#D = DEPARTMENT.#D AND
GROUP.#Curator = TEACHER.#T AND Name = 'ТП' AND COURSE = 1.

Запит 5. Вивести список лекцій, на яких кількість студентів в групі перевищує кількість місць в аудиторії. У списку вказати номер аудиторії, номер групи, лекційну дисципліну, тиждень, день тижня:

SELECT ROOM.Num, GROUP.Num, SUBJECT.Name, Week, Day
FROM LECTURE, GROUP, SUBJECT, ROOM
WHERE LECTURE.#G = GROUP.#G AND LECTURE.#S = SUBJECT.#S
AND LECTURE.#R = ROOM.#R AND
GROUP.Quantity > ROOM.Seats.

5.2.4. Використання агрегатних функцій. Функції в мові SQL дозволяють виконувати різні обчислювальні операції, які не можливо виразити стандартними засобами мови. Вони дозволяють істотно збільшити можливості з маніпулювання даними. Стандарт ANSI визначає набір функцій, проте в конкретних СУБД цей список, як правило, істотно розширюється. Наприклад, SQL Oracle містить більше 120 функцій.

Із безлічі наявних функцій ми розглянемо тільки, так звані, агрегатні функції, які включені в стандарт ANSI. До них відносяться такі:

- COUNT – повертає кількість значень вказаного стовпця;.
- SUM – повертає суму всіх значень стовпця;
- AVG – повертає середнє значення всіх значень стовпця;
- MAX – повертає найбільше значення в стовпці;

- MIN – повертає найменше значення в стовпці.

Крім спеціального випадку COUNT (*), кожна з цих функцій оперує сукупністю значень стовпця деякої таблиці і створює єдине значення.

Для функцій SUM і AVG цей стовпець повинен містити числові значення.

Слід відмітити, що тут стовпець – це стовпець віртуальної таблиці, в якому можуть міститися дані не тільки із стовпця базової таблиці, але і дані, отримані шляхом функціонального перетворення. При цьому вираз, що визначає стовпець таблиці, може бути скільки завгодно складним, але не повинен містити SQL-функцій (вкладеність SQL-функцій не допускається). Наприклад, вираз типу **AVG (Ставка+Надбавка/2)** є цілком допустимим. Аргументу всіх функцій, окрім COUNT (*), може передувати ключове слово DISTINCT. Спеціальна ж функція COUNT (*) служить для підрахунку всіх без виключення рядків в таблиці (включаючи дублювання).

Ці функції можна застосовувати тільки після ключового слова SELECT. Якщо в запиті немає ключового слова GROUP BY, то область дії агрегатної функції розповсюджується на все результуюче відношення, а якщо це ключове слово присутнє, то на створювані нею групи. Розглянемо на прикладах використання цих функцій.

1. Вивести кількість кафедр на факультеті інформатики:

```
SELECT  COUNT (*)
FROM      FACULTY ,  DEPARTMENT
WHERE   FACULTY.F = DEPARTMENT.#F AND
FACULTY.Name = 'IT'.
```

2. Вивести кількість предметів, що читаються у вузі:

```
SELECT  COUNT (*)  Number-Of-Subjects FROM  SUBJECT.
```

3. Вивести місткість всіх аудиторій у корпусі 15:

SELECT SUM(Seats) Total_Number-Of-Seat-In-Building-15

FROM ROOM

WHERE Building = 15.

4. Вивести кількість студентів на факультеті інформатики:

SELECT SUM(GROUP. Quantity)

Number-Of-Students-In-IT-Faculty

FROM FACULTY , DEPARTMENT, GROUP

WHERE FACULTY.#F = DEPARTMENT.#D AND DEPARTMENT.#D = GROUP.#D AND FACULTY.Name = 'IT'.

5. Вивести найбільший фонд серед кафедр факультету інформатики:

SELECT MAX (DEPARTMENT.Fund)

MAX- DEPARTMENT-Fund-En-IT-Faculty

FROM FACULTY , DEPARTMENT

WHERE FACULTY.#F = DEPARTMENT.#F AND FACULTY.Name = 'IT'.

6. Вивести місткість всіх аудиторій корпусу 15, кількість місць в найменшій і найбільшій аудиторіях, середню місткість:

SELECT SUM(Seats) Total_Number-Of-Seats-In-Building-15

MIN(Seats) Seats-In-The-Smallest-Room

MAX(Seats) Seats-In-The-Largest-Room

AVG(Seats) / Count(*) Average-Number-Of-Seats

FROM ROOM WHERE Building = 15.

5.2.5. Ключове слово GROUP BY. Групування таблиці по рядках. Ключове слово GROUP BY дозволяє об'єднати безліч рядків, що отримуються після застосування ключового слова WHERE, в групи по ознаці рівності значень одного або декількох стовпців. В цьому випадку агрегатні функції, використовувані у ключовому слові SELECT, діють не на всьому результуючому відношенні, а в межах кожної групи. Далі, якщо в ключовому слові SELECT, присутні імена стовпців, то вони

повинні бути ті стовпці, за допомогою яких проводиться групування. Розглянемо приклад.

Вивести на екран кількість студентів на всіх кафедрах факультету інформатики:

SELECT D.Name, SUM (G.Quantity) Count-Of-Students FROM FACULTY F, DEPARTMENT D, GROUP G WHERE F.#F=D.#D AND D.#D = G.#D AND F.Name='IT' GROUP BY D.Name.

5.2.6. Ключове слово HAVING. Завдання умов вибірки на групах рядків. Ключове слово HAVING виконує таку ж роль для груп, що і ключове слово WHERE для рядків: воно дозволяє задавати умови для рядків, що формуються ключовим словом GROUP BY. Це ключове слово включається в пропозицію лише за наявності ключового слова GROUP BY, а вираз в HAVING повинен приймати єдине значення для групи. У формульованих в цьому ключовому слові умовах можна використовувати агрегатні функції, що діють в межах створюваних груп, а також використовувати ті стовпці, по яких проводиться групування. Розглянемо приклади.

1. Вивести ті факультети, у яких фонд факультету перевищує сумарний фонд всіх кафедр

SELECT F.Name FROM FACUTY F, DEPARTMENT D WHERE F.#F=D.#F GROUP BY F.Name HAVING F.Fund > SUM (D.Fund).

2. Вивести ті факультети, у яких фонд факультету перевищує сумарний фонд всіх кафедр на 2000:

SELECT F.Name FROM FACUTY F, DEPARTMENT D WHERE F.#F=D.#F GROUP BY F.Name HAVING (F.Fund – SUM (D.Fund))> 2000.

5.2.7. Ключове слово ORDER BY. Впорядкування результуючих рядків. Ключове слово має очевидний сенс, в ньому перераховуються стовпці результуючого відношення, за якими необхідно

відсортувати вихідні рядки, а також порядок їх сортування: за збільшенням – ASC або за зменшенням – DESC. Слід відмітити, що сортувати можна за будь-якими стовпцями, які присутні в таблицях, що сполучаються, а не обов'язково за тими, які приводяться у ключовому слові SELECT.

1. Вивести імена викладачів вузу, впорядкованими за зменшенням (імен):

SELECT Name FROM TEACHER ORDER BY Name DESC.

2. Вивести кількість викладачів на всіх кафедрах вузу, упорядкувати їх за кількістю викладачів:

SELECT T.Name, Count (*) Number-Of-Teacher-In-The-Faculty
FROM DEPARTMENT D, TEACHER T
WHERE D.#D = T.#D GROUP BY D.Name
ORDER BY Number- Of-Teacher-In-The-Faculty.

5.2.8. Порядок обчислення ключових слів. В запиті ключові слова слідують в наступному порядку: SELECT, FROM, WHERE, GROUP BY, HAVING, ORDER BY. Їх використання при виконанні запиту можна представити таким чином:

- проводиться декартовий добуток рядків з таблиць, вказаних у ключовому слові FROM;
- до отриманої єдиної таблиці застосовуються умови з ключового слова FROM, ці умови формулюються таким чином, що їх істинність перевірялася на рядку таблиці;
- отримані рядки групуються згідно умові у ключовому слові GROUP BY;
- до згрупованих рядків застосовуються умови, задані у ключовому слові HAVING, вибираються тільки ті групи, які задовольняють ці умови;
- рядки (групи) упорядковуються згідно ключового слова ORDER

BY;

- виводяться ті стовпці або вирази, які присутні у ключовому слові SELECT.

5.2.9. Підзапити: вкладання блоків SELECT один в інший. *Вкладений підзапит* - це запит, результат виконання якого передається як аргумент в інший запит. Способом вкладання підзапиту в запит є його використання у ключовому слові WHERE або HAVING в правому аргументі одного з наступних предикатів: IN, EXISTS =, <>, <,| <=,| >,| >=.

Існують прості (незалежні) і корельовані (залежні) вкладені підзапити.

Простий (незалежний) вкладений підзапит – це такий підзапит, обчислення якого відбувається незалежно від обчислення зовнішнього запиту. Такі запити обробляються системою "знизу вгору". Першим оброблюється вкладений підзапит. Безліч значень, отримана в результаті його виконання, використовується при реалізації зовнішнього підзапиту.

Корельваний (залежний, зв'язаний) вкладений підзапит - це такий підзапит, обчислення якого залежить від процесу обчислення в зовнішньому запиті. Такі запити обробляються системою в зворотному порядку. Спочатку вибирається поточний рядок з таблиці зовнішнього запиту і на підставі значень її полів виконується обчислення підзапиту (тобто в умові обчислення підзапиту присутні значення полів з зовнішнього запиту). Потім перевіряється умова WHERE на включення поточного рядка зовнішнього запиту в результат.

*Повернення одного значення***.** При використанні предикатів, що порівнюють два значення (=, ≠, <, ≤, >,≥), підзапит повинен повертати одне значення. Інакше видається повідомлення про помилку. Наведемо приклади.

1. Вивести список кафедр вузу, які розташовуються в тому ж корпусі, що і кафедра ТП:

SELECT Name FROM DEPARTMENT WHERE Building =
(SELECT Building FROM DEPARTMENT WHERE Name='ТП'*).*

Внутрішній запит визначає корпус кафедри 'ТП'; він використовується в предикаті порівняння (=) зовнішнього запиту. В підзапиту можна використовувати агрегатну функцію, що гарантує повернення єдиного значення.

2. Вивести факультети, фонд яких перевищує фонд всіх кафедр факультету інформатики:

SELECT Name FROM FACULTY
WHERE Fund > SELECT SUM (D.Fund)
FROM FACULTY F , DEPARTMENT D

*Повернення багатьох значень***.** Якщо використовується предикат, перевіряючий входження (IN) або не входження (NOT IN) окремого значення в множину (або деякі інші, про яких піде мова далі), то підзапит може повертати безліч значень. Продемонструємо це на прикладі.

Вивести хто з викладачів факультету інформатики також працює на інших факультетах:

SELECT T.Name FROM FACULTY F, DEPARTMENT D,
TEACHER T WHERE F.#F=D.#F AND D.#D=T.#D AND
F.Name='КНiЕК' AND T.Name IN (SELECT T.Name FROM
FACULTY F, DEPARTMENT D, TEACHER T WHERE F.#F=D.#F
AND D.#D=T.#D AND F.Name!='IT').

*Корелювання (поєднання) підзапиту із запитом***.** В деяких випадках характер обчислення підзапиту залежить від значення поточного рядка зовнішнього запиту.

Обробка корельованого підзапиту, відповідно, повинна повторюватися для кожного значення, яке отримується із зовнішнього підзапиту, а не виконуватися раз і назавжди як в не зв'язаному підзапиту.

Розглянемо декілька прикладів такого поєднання.

1. Вивести факультети, у яких фонд фінансування менший, ніж сума фондів фінансування всіх їх кафедр:

SELECT Name

FROM FACULTY WHERE Fund <

(SELECT(Fund) FROM DEPARTMENT

WHERE DEPARTMENT.#F = FACULTY.F).

2. Вивести на екран викладачів, які не є кураторами груп:

SELECT Name FROM TEACHER WHERE NOT EXISTS

(SELECT * FROM GROUP WHERE TEACHER.#T =

GROUP.#Curator).

Підзапит у ключовому слові HAVING. Як вже наголошувалося, підзапити можуть вкладатися і у ключове слово HAVING. Розглянемо приклад.

Вивести назви кафедр і кількість студентів на них для тих кафедр, кількість студентів на яких більша, ніж на кафедрі і технологій програмування:

SELECT D.Name, SUM (G.Quantity) FROM DEPARTMENT D,

GROUP G WHERE D.#D=G.#D GROUP BY D HAVING SUM

(G.QUANTITY) > (SELECT SUM(G.Quantity) FROM DEPARTMENT

D, GROUP G WHERE D.#D=G.#D AND D.Name = 'ТП').

5.2.10. Використання предикатів *ANY, ALL, EXISTS і IN.* Предикати ANY I ALL слідують за одним із предикатів =, !=, >, <, <=, >= і перевіряє, чи виконується цей предикат хоча б для одного (при ANY) або для всіх (при ALL) значеннях множини, заданої в правій частині, по відношенню до елементу, заданого у лівій частині.

Наприклад, для отримання списку кафедр, фонди яких більші хоча б однієї з кафедр факультету інформатики, слід записати:

SELECT Name FROM DEPARTMENT D

WHERE Fund > ANY (SELECT D.Fund FROM FACULTY F, DEPARTMENT D

WHERE F.#F = D.#F AND F.Name = 'IT').

Якщо ж необхідно, щоб фонди кафедр були більше будь-якої (тобто всіх) з кафедр факультету інформатики, то це записується таким чином:

SELECT Name

FROM DEPARTMENT

WHERE Fund > ALL (SELECT DEPARTMENT.Fund

FROM FACULTY, DEPARTMENT

WHERE FACULTY.#F = DEPARTMENT.#F AND

FACULTY.Name = 'IT').

EXISTS є одномісним предикатом, який повертає TRUE, якщо підзапит, до якого він застосовується, містить хоча б один рядок. Наприклад, для отримання списку викладачів, які читають хоча б один курс лекцій, слід записати:

SELECT Name FROM TEACHER WHERE EXISTS

(SELECT * FROM LECTURE WHERE LECTURE.#T =

TEACHER.#T).

Якщо ж нас цікавлять ті викладачі, які не читають ніяких лекцій, то слід записати:

SELECT Name FROM TEACHER WHERE NOT EXISTS

(SELECT * FROM LECTURE WHERE LECTURE.#T =

TEACHER.#T).

Предикат IN перевіряє входження елемента в множину. Лівий операнд предиката повинен бути специфікацією одного значення, наприклад, константа або ім'я поля, а правий операнд – множиною, наприклад, константа-множина або select-запит. Константа-множина специфікується списком елементів множини, які беруться у фігурні

дужки, наприклад {'Іванов', 'Петров', 'Ігнатов'}. Розглянемо ряд прикладів.

1. Вивести назви факультетів, розташованих в корпусах 1, 3, 11, 15:

SELECT Name FROM FACULTY

WHERE Building IN {1,3,11,15}.

2. Вивести на екран назви факультетів, розташованих в тих самих корпусах, що і факультети інформатики і економіки:

SELECT Name FROM FACULTY

WHERE Building IN (SELECT Building FROM FACULTY

WHERE Name = 'Інформатика' OR Name = 'Економіка').

3. Вивести факультети, розташовані в корпусах, що відрізняються від тих, в яких розташовані факультети інформатики і економіки:

SELECT Name FROM FACULTY WHERE Building NOT IN

(SELECT Building FROM FACULTY

WHERE Name = 'Інформатика' OR Name= 'Економіка')

5.2.11. Використання теоретико-множинних операторів. Мова SQL має три теоретико-множинні оператори: UNION, INTERSECT, MINUS. Вони дозволяють об'єднувати, перетинати і отримувати різність двох множин. Аргументами цих операторів є відношення, отримані в результаті виконання пропозицій SELECT. Таблиці-операнди повинні бути сумісними по теоретико-множинним операціям. Це означає, що вони повинні мати однакову кількість стовпців і відповідні пари стовпців повинні мати сумісний тип даних. Розглянемо декілька прикладів.

1. Вивести факультети, розташовані в корпусах 5 і 6:

SELECT Name FROM FACULTY WHERE Building = 5

INTERSECT

SELECT Name FROM FACULTY WHERE Building = 6.

Відзначимо, що операція перетину може бути виражена також за допомогою предиката IN таким чином :

SELECT Name FROM FACULTY WHERE Building = 5 AND NAME IN (SELECT Name FROM FACULTY WHERE Building = 6).

2. Вивести імена викладачів, що викладають лекції з дисципліни "Бази даних", але які не викладають лекції з дисципліни "Програмування":

SELECT T.Name FROM TEACHER T, LECTURE L, SUBJECT S

WHERE T.#T = L.T AND L.#S = S.#S AND S.Name = 'Бази даних'

MINUS

SELECT T.Name FROM TEACHER T, LECTURE L, SUBJECT S

WHERE T.#T = L.T AND L.#S = S.#S AND S.Name = 'Програмування'

Необхідно зазначити, що операція перетину може бути виражена також за допомогою предиката NOT IN таким чином:

SELECT T.Name FROM TEACHER T, LECTURE L, SUBJECT S

WHERE T.#T = L.T AND L.#S = S.#S AND S.Name = 'Бази даних' AND

T.Name NOT IN (SELECT T.Name FROM TEACHER T, LECTURE L, SUBJECT S WHERE T.#T = L.T AND L.#S = S.#S AND S.Name = 'Програмування').

3. Вивести список імен всіх деканів і завідувачів кафедр:

SELECT Name FROM FACULTY UNION

SELECT Name FROM DEPARTMENT.

5.2.12. Використання невизначених значень. Якщо при завантаженні даних не введено значення в будь-яке поле таблиці, то

СУБД розмістить в нього NULL-значення. Аналогічне значення можна ввести в поле таблиці, виконуючи операцію змінення даних. Мова SQL надає можливість маніпулювати невизначеними значеннями. При цьому використовується логіка, в якій саме NULL-значення не вважається рівним іншому NULL-значенню. У зв'язку з цим, щоб врахувати в запитах умови на невизначені значення, введені предикати IS, IS NOT, які можна використовувати з константою NULL. Наприклад, якщо необхідно підрахувати середній фонд фінансування кафедр за умови, що фонд дійсно заданий, то слід записати наступний запит:

```
SELECT  AVG (Fund)
FROM    DEPARTMENT
WHERE   Fund IS NOT NULL.
```

Якщо ж необхідно вивести імена кафедр, для яких не заданий фонд фінансування, то слід записати:

```
SELECT  Name
FROM    DEPARTMENT
WHERE   Fund IS NOT NULL.
```

5.3. Засоби маніпулювання даними

До цих пір ми вивчали, яким чином вибираються дані з бази даних. Після вибірки дані можуть використовуватися користувачами для їх аналізу, або прикладною програмою для їх обробки. Природно виникає питання, а як можна дані поміщати в базу даних, редагувати їх і навіть видаляти. Далі ми розглянемо пропозиції SQL, які дозволяють маніпулювати даними. Цими пропозиціями є:

- INSERT;
- UPDATE;
- DELETE.

5.3.1. Вставка рядків в таблицю. Пропозиція INSERT. Пропозиція

INSERT дозволяє вводити дані в базу даних. Є два різновиди цієї пропозиції:

- INSERT...VALUES;
- INSERT...SELECT.

Пропозиція INSERT...VALUES дозволяє вставити в таблицю один рядок. Його синтаксис такий:

INSERT INTO table-name (col1, col2…) VALUES (value1, value 2…).

При цьому слід дотримуватися наступних правил:

- дані, що вставляються, повинні узгоджуватися з типами даних відповідних стовпців;
- розміри даних повинні відповідати розмірам стовпців;
- порядок даних у ключовому слові VALUES повинен відповідати порядку перерахування стовпців.

Наведемо приклади.

1. Вставити рядок в таблицю FACULTY:

INSERT INTO FACULTY (#F, Name, Dean, Building, Fund)

VALUES (15, ‘IT’, ‘Коваленко Д.О.’, 5, 25000).

Список імен стовпців не обов'язковий, якщо вставляються значення всіх стовпців. У цьому випадку порядок проходження значень повинен співпадати з порядком стовпців у відношенні.

2. Вставити рядок в таблицю DEPARTMENT:

INSERT INTO DEPARTMENT

VALUES (03,15, ‘ТП’, ‘Литвиненко О.В.’, 5, 5500).

Якщо значення яких-небудь стовпців не відомі, то слід використовувати константу NULL.

3. Вставити рядок в таблицю TEACHER:

INSERT INTO TEACHER

VALUES (173, 13, ‘Резніченко Є.П.’, NULL, ‘266-18-15’).

Значення NULL не можуть приймати ті стовпці, які є обов'язковими.

Багато СУБД дозволяють створювати опис стовпців з атрибутом UNIQUE. Він означає, що в межах таблиці значення такого стовпця повинні бути такими, що не повторюються. У зв'язку з цим при вставці рядків можуть бути відмови, якщо це обмеження порушується.

Пропозиція INSERT…SELECT. Ця пропозиція дозволяє вставити в таблицю безліч рядків, які є результатом виконання запиту. Вона дозволяє копіювати інформацію з однієї або декількох таблиць в іншу. Такі "похідні" таблиці створюються на користь підвищення продуктивності виконання тих або інших операцій над базою даних. Загальний формат пропозиції такий:

INSERT INTO table_name (col1, col2…)

SELECT col1, col2…FROM tablename WHERE

search_condition.

Істотним є те, що вихідні результати стандартної пропозиції SELECT є вхідними даними пропозиції INSERT.

Пропозиція INSERT...SELECT повинна задовольняти таким додатковим правилам:

- пропозиція SELECT не може вибирати рядка з таблиці, в яку проводиться вставка;
- кількість стовпців у ключовому слові INSERT INTO повинна співпадати з кількістю стовпців у ключовому слові SELECT;
- типи даних стовпців у ключовому слові INSERT INTO повинні співпадати з типами даних стовпців у ключовому слові SELECT.

5.3.2. Оновлення даних. Пропозиція UPDATE. Мета пропозиції полягає у зміні значення існуючих рядків. Її синтаксис такий:

UPDATE table_name

SET column_name_l = expression_1[, colum_name_2 =

expression_2]…

WHERE search_condition.

Оновлення за умовою. Всі рядки таблиці, що задовольняють умові, заданій у ключовому слові WHERE, змінюються згідно ключовому слову SET. Розглянемо приклад.

Встановити фонд факультету інформатики рівним 250300:

UPDATE FACULTY SET Fund = 250300

WHERE Name = ‘IT’.

Оновлення безумовне. Якщо фраза WHERE не задана, то оновленню піддаються всі рядки. Розглянемо приклад.

Всім факультетам встановити фонд 260500:

UPDATE FACULTY SET Fund = 260500.

5.3.3. Видалення рядків таблиці. Пропозиція DELETE. Пропозиція DELETE проводить видалення рядків таблиці. Її синтаксис такий:

DELETE FROM table_name

WHERE condition.

В залежності від використання ключового слова WHERE можливо:

- видалити один рядок;
- видалити безліч рядків;
- видалити всі рядки;
- не видалити жодного рядку.

Приведемо декілька специфічних особливостей використання пропозиції DELETE.

- Пропозиція DELETE не дозволяє видаляти окремі поля (використовуйте для цього UPDATE). Пропозиція DELETE видаляє повністю весь рядок.
- Як і пропозиції INSERT і UPDATE, вона може викликати

проблему порушення цілісності**.** Пам'ятаєте про цю проблему в процесі маніпулювання даними в базі даних.

- Якщо у ключовому слові WHERE використовується вкладений підзапит, то у ключовому слові FROM цього підзапиту не повинно згадуватися у таблиці, з якої видаляються рядки. Це відноситься і до пропозицій INSERT, UPDATE.
- Пропозиція DELETE видаляє рядки таблиці, а не саму таблицю. Для видалення всієї таблиці використовується пропозиція DROP TABLE. Розглянемо приклади.

1. Видалити відомості про лекції, які проводяться в суботу і неділю:

DELETE FROM LECTURE WHERE Day IN (‘Субота’, ‘Неділя’).

2. Видалити всі рядки із таблиці SUBJECT:

DELETE FROM SUBJECT.

3. Видалити всі предмети з таблиці SUBJECT, які не використовуються при читанні лекцій:

DELETE FROM SUBJECT WHERE #S != ALL (SELECT #S FROM LECTURE).

Контрольні питання

1. Створити запит вибірки з декількох таблиць.
2. Створити запит вибірки всіх рядки з однієї таблиці.
3. Як згрупувати таблиці по рядках?
4. Як видалити рядки з таблиці?

6 ЛАБОРАТОРНИЙ ПРАКТИКУМ

6.1 Учбовий проект – постановка завдання

В учбовому проекті розглядається база даних центру сертифікації рослин. Далі наводяться описи даних, які збираються, супроводжуються та використовуються співробітниками цього центру для визначення того, чи може заявлений сорт бути занесений до реєстру сортів.

Вимоги до даних

Культури

Сертифікаційним центром можуть бути прийняті заявки на нові сорти різних культур. Кожна культура характеризується кодом, назвою українською і латинською мовами, та представлена зображенням.

Сорти

Кожний сорт, який проходить випробування, характеризується кодом сорту, назвою сорту та кодом культури, до якої він належить.

Кліматична зона

Випробування відбуваються у різних кліматичних зонах України. У свою чергу, кожна кліматична зона характеризується своїм кодом, назвою та описом природних умов.

Станції

Польові випробування нових сортів відбуваються на дослідних станціях, кожна з яких характеризується кодом, назвою та кодом кліматичної зони, в якій розташована станція. Крім того, зазначається адреса і телефон станції, прізвище, ім’я та по батькові директора, загальна площа земельних угідь.

Результати випробувань

Дані польових випробувань для кожного заявленого сорту представлені параметрами, назви яких залежать від культури, до якої належить сорт. В учбовому проекті будемо розглядати лише один

спільний для всіх культур показник – врожайність сорту. Таким чином, для кожного випробування має міститися така інформація: код станції, код культури, код сорту, рік випробування та врожайність.

Вимоги до операцій

Операція 1

Створення і супроводження записів з інформацією щодо сільськогосподарських культур.

Операція 2

Введення нової заявки на новий сорт рослин – в результаті з'являється новий запис з інформацією щодо сортів.

Операція 3

Створення і супроводження записів з інформацією щодо дослідних станцій.

Операція 4

Введення даних дослідних випробувань.

Операція 5

Швидкий пошук необхідної інформації

Операція 6

Аналіз даних польових випробувань – в результаті визначається, чи може заявлений сорт бути занесений до реєстру сортів.

Операція 7

Формування звітної документації

Завдання на виконання лабораторної роботи

1. Побудуйте структуру бази даних як сукупність таких таблиць:

- Культура
- Сорт
- Кліматична_зона
- Станція

- Випробування

2. Для кожної таблиці визначте стовпчики відповідно постановці завдання, для кожного стовпчика – тип та інші необхідні обмеження, для кожної таблиці – первинний ключ.

3. Побудуйте схему (діаграму) бази даних, вказуючи зв’язки між таблицями та встановлюючи умови цілісності.

4. Подати звіт із побудованою схемою.

Приклад оформлення звітів

Створена таблиця «Культура» в режимі конструктора представлена на рис.6.1.

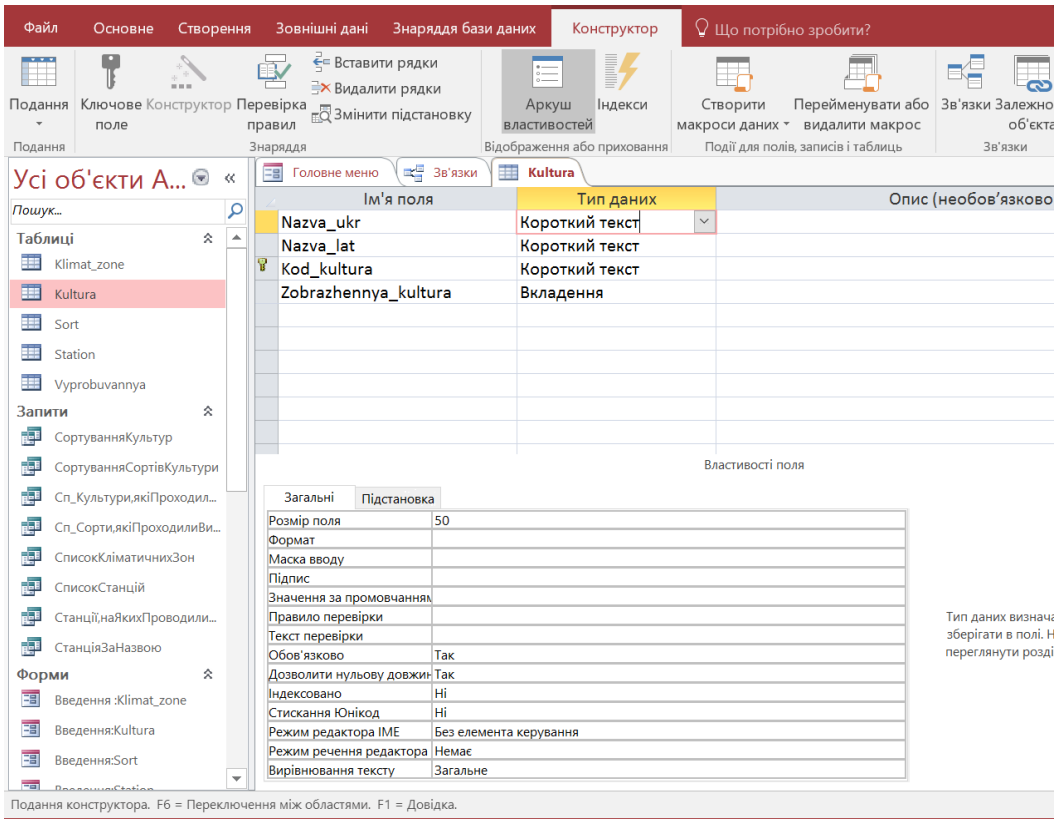


Рис.6.1 Таблиця «Культура» в режимі конструктора

Створена таблиця «Сорт культури» в режимі конструктора представлена на рис.6.2.

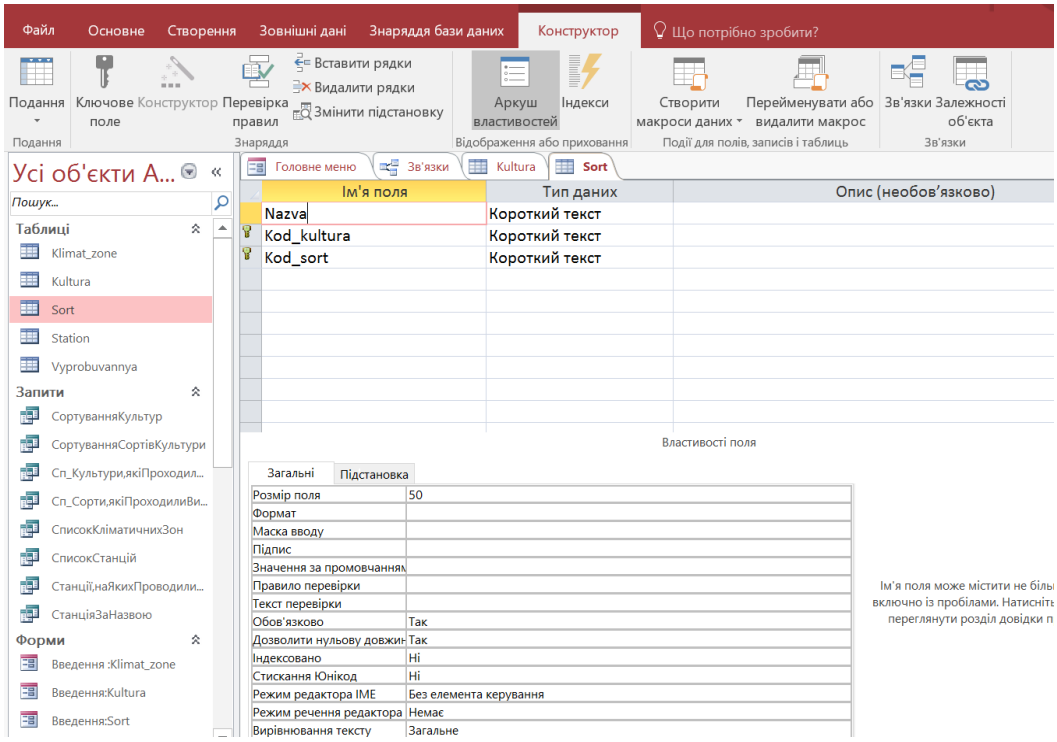


Рис.6.2 Таблиця «Сорт культури» в режимі конструктора

Створена таблиця «Кліматична зона» в режимі конструктора представлена на рис.6.3.

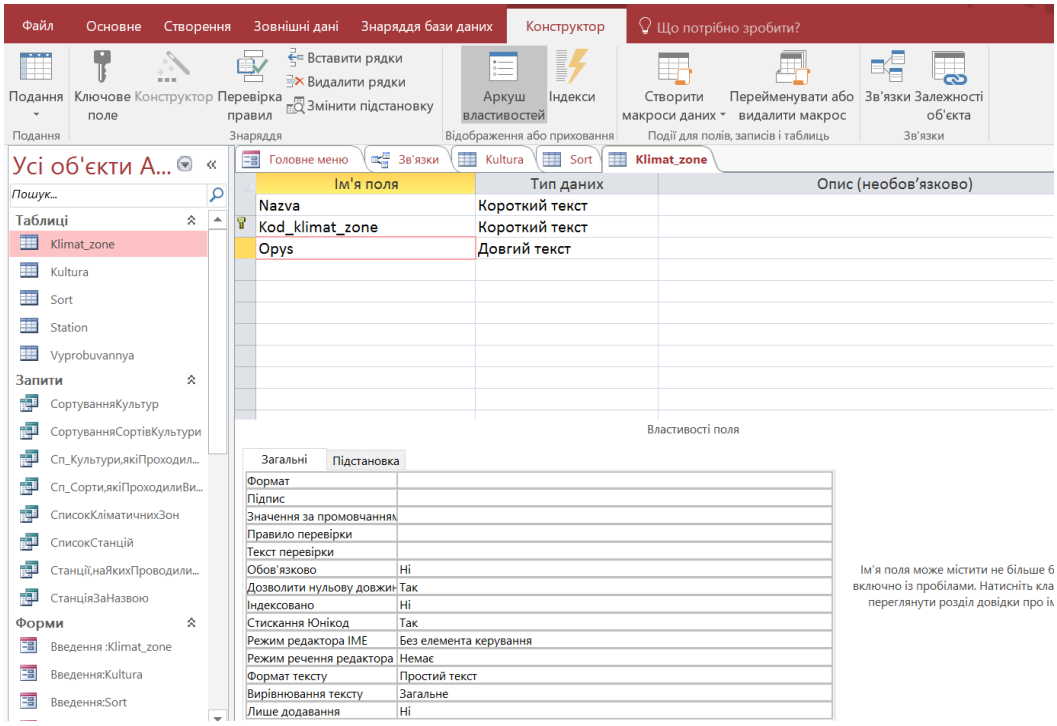


Рис.6.3 Таблиця «Кліматична зона» в режимі конструктора

Створена таблиця «Станція» в режимі конструктора представлена на рис.6.4.

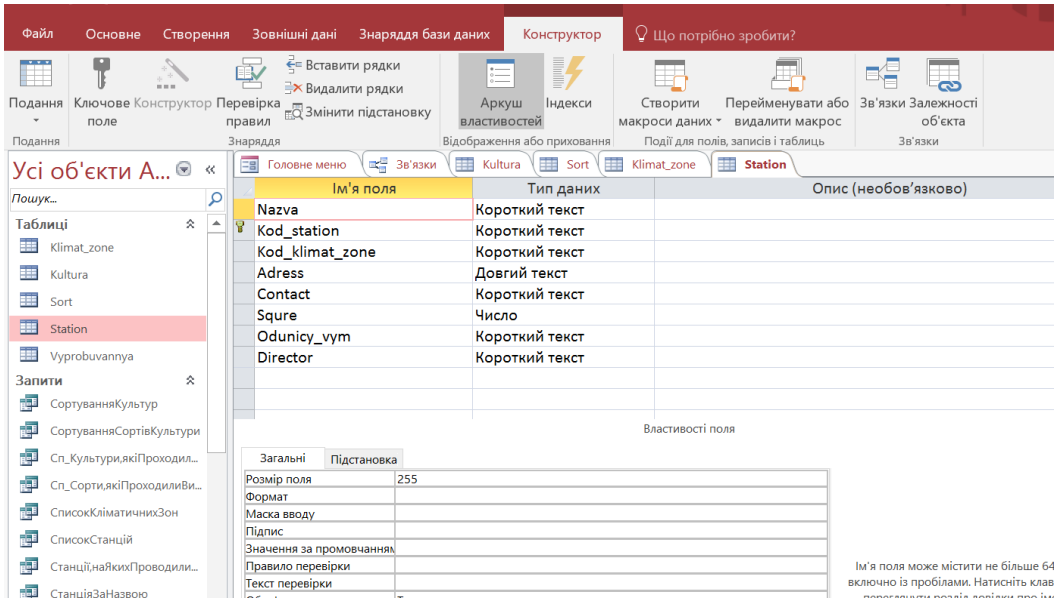


Рис.6.4 Таблиця «Станція» в режимі конструктора

Створена таблиця «Випробування» в режимі конструктора представлена на рис.6.5.

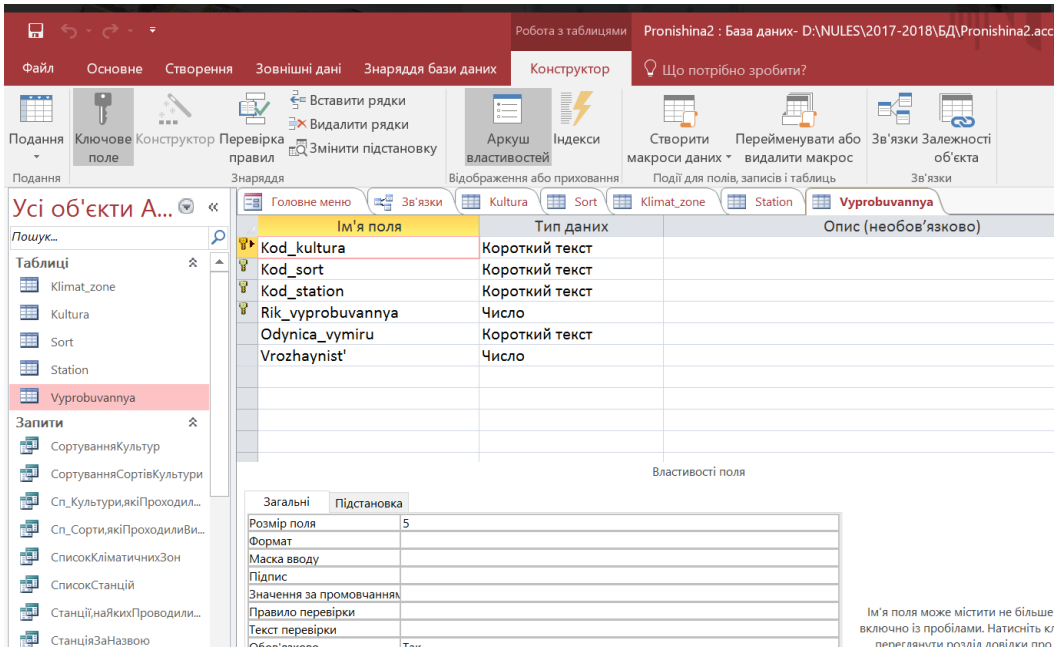


Рис.6.5 Таблиця «Випробування» в режимі конструктора

Схема бази даних представлена на рис.6.6.

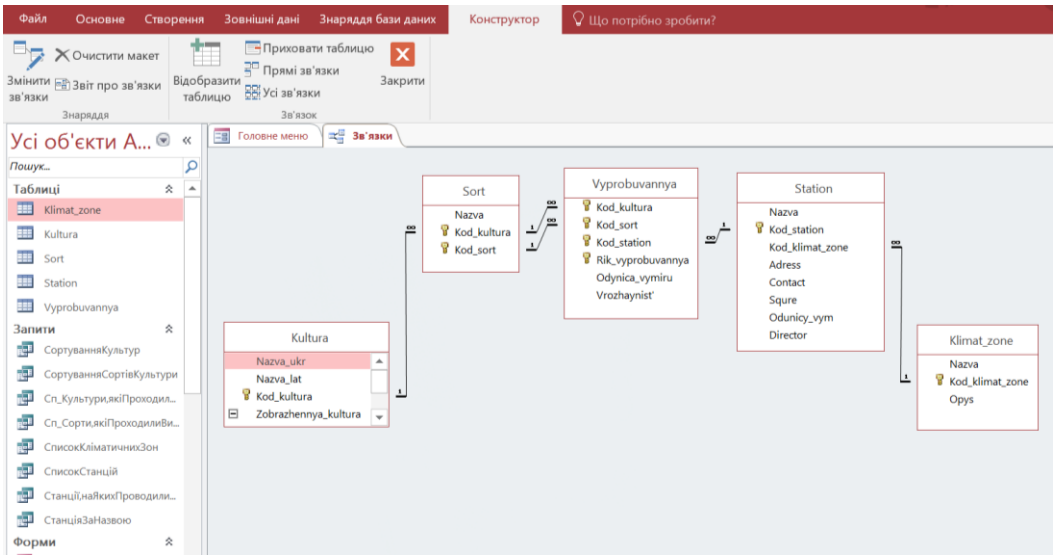


Рис.6.6 Схема бази даних

6.2 Створення форм введення даних

Теоретична частина

Форма – це об’єкт бази даних Access, який містить шаблон інтерфейсу користувача бази даних та певні алгоритми обробки інформації. Оскільки робота з базою даних призначена для кінцевого користувача, форми мають надати йому можливість у зручному та зрозумілому вигляді виконувати усі необхідні функції, що покладені на нього.

Умовно усі форми можна розділити на два типи: такі, що не прив’язані до таблиць (так звані, кнопкові форми), та такі, що безпосередньо надають можливість працювати з даними бази.

Форми першого типу використовують для надання користувачеві простого способу переходу по різних вікнах (формах) проекту. Відповідно вони складаються з кнопок, при натисканні на які відбуваються певні дії – відкриваються чи закриваються форми. Усі кнопки мають супроводжуватися спливаючими підказками. Крім кнопок,

такі форми можуть містити і інші елементи управління, наприклад, написи, рисунки, перемикачі, прапорці тощо.

Форми другого типу складаються із будь-яких елементів управління, серед яких обов’язково присутнім є елемент «поле» або «поле із списком». Ці елементи надають можливість отримати доступ безпосередньо у таблицю бази даних.

Для створення кнопкової форми достатньо скористатися режимом конструктора. Створення форм другого типу легко здійснити у режимі майстра. Після створення шаблону форми у режимі майстра усі необхідні зміни виконуються у режимі конструктора.

Постановка завдання

Основна мета лабораторної роботи – створення інтерфейсу користувача для введення даних в усі таблиці бази та тестування отриманих інтерфейсів шляхом внесення інформації.

1. Внесення інформації щодо культур.

Форма має містити усі необхідні елементи управління для безпосереднього внесення даних про кожну культуру. Передбачити спливаючі підказки для елементу управління «поле» та розмістити на формі кнопки переходу по записах.

2. Внесення інформації щодо заявлених сортів.

Форма має містити усі необхідні елементи управління, та для вибору культури (коду культури) використати елемент управління «поле із списком». Цей елемент має складатися з двох стовпчиків – коду культури і назви культури українською мовою. Передбачити спливаючі підказки для елементів управління «поле» і «поле із списком», та розмістити на формі кнопки переходу по записах.

3. Внесення інформації щодо кліматичних зон України.

Форма має містити усі необхідні елементи управління для безпосереднього внесення даних про усі кліматичні зони України.

Передбачити спливаючі підказки для елементу управління «поле» та розмістити на формі кнопки переходу по записах.

4. Внесення інформації щодо дослідних станцій.

Форма має містити усі необхідні елементи управління, та для вибору кліматичної зони (її коду) використати елемент управління «поле із списком». Цей елемент має складатися з двох стовпчиків – коду і назви кліматичної зони. Передбачити спливаючі підказки для елементі управління «поле» і «поле із списком», та розмістити на формі кнопки переходу по записах.

5. Внесення результатів польових випробувань.

Для вибору станції, на якій відбуваються випробування, культури та відповідного сорту використати елемент управління «поле із списком». Передбачити виведення назв станцій, культур та сортів. У поле, яке призначено для внесення року випробування, вказати за замовченням значення поточного року. Розмістити на формі кнопки переходу по записах та усі необхідні спливаючі підказки.

Завдання на виконання лабораторної роботи

1. Розробити форми введення інформації в усі таблиці бази даних.

2. Для тестування занести:

a. п’ять культур;

b. по три сорти для кожної культури;

c. усі необхідні кліматичні зони України;

d. по три дослідні станції для кожної кліматичної зони;

e. для кожного сорту передбачити внесення результатів випробування по кожній кліматичній зоні від одного до п’яти років.

У звіті представити скрін-шоти усіх форм та перелік внесеної інформації.

Приклад оформлення звітів

Створена форма «Культура» в режимі конструктора представлена на рис. 6.7.

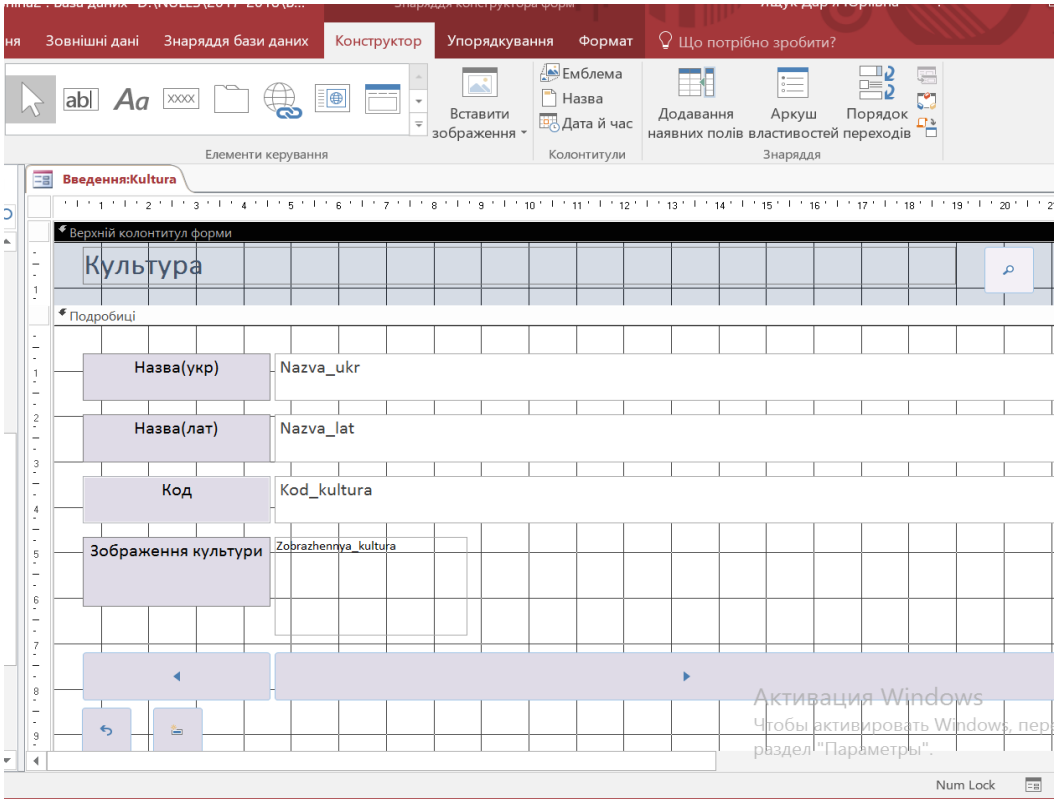


Рис.6.7 Форма «Культура» в режимі конструктора

Форма для введення даних про культуру зображена на рис.6.8.

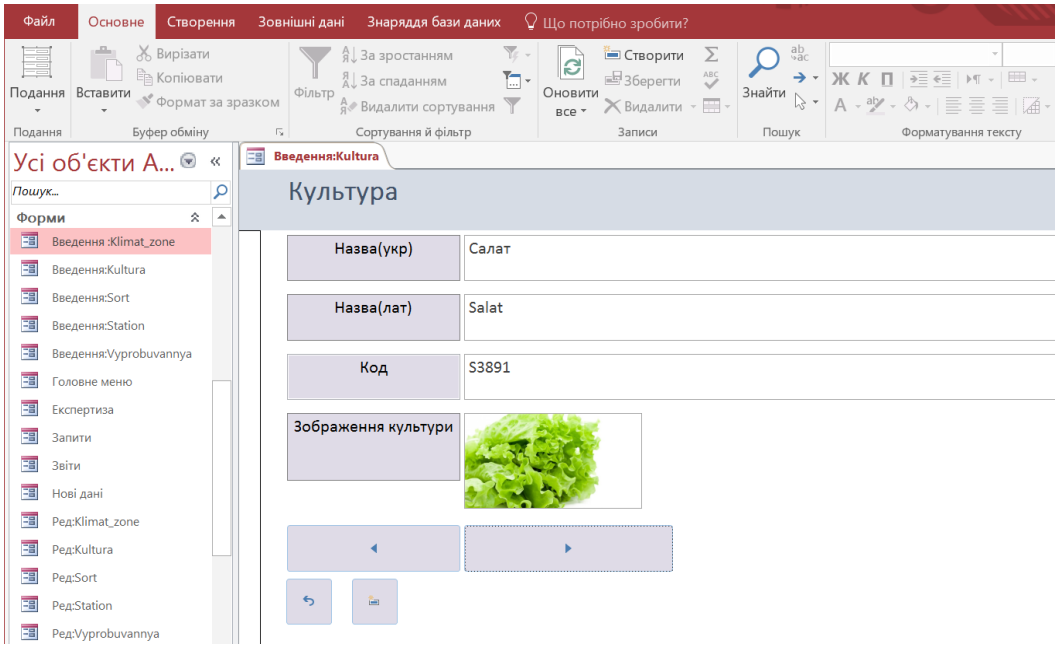


Рис.6.8 Форма «Культура»

Створена форма «Сорт культури» в режимі конструктора представлена на рис. 6.9.

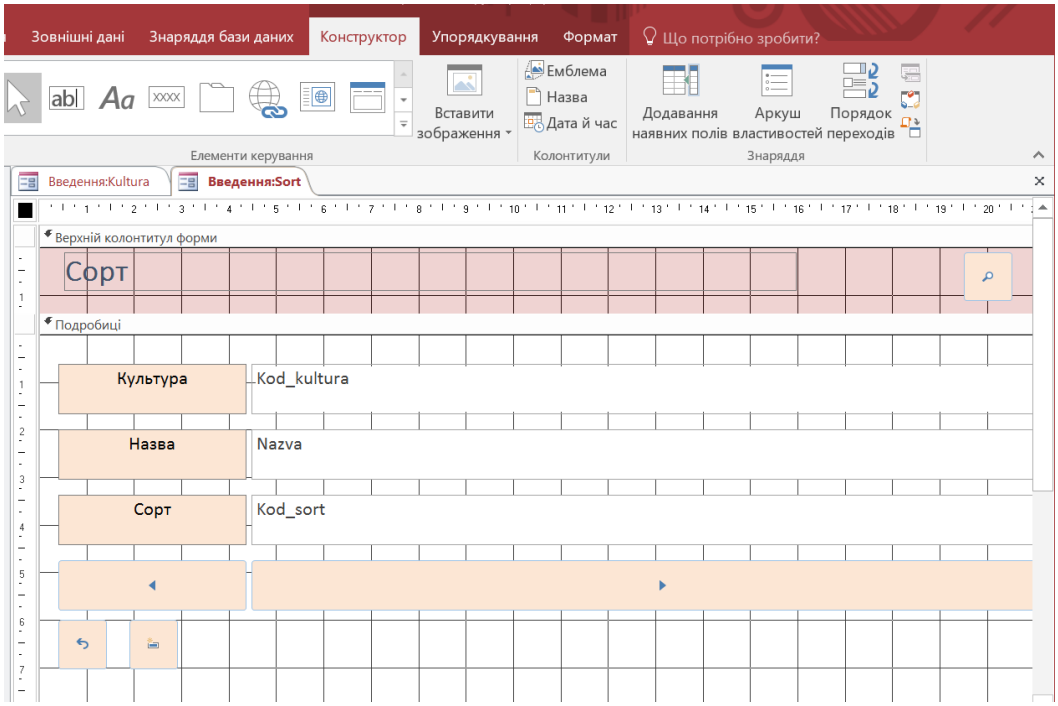


Рис.6.9 Форма «Сорт культури» в режимі конструктора

Форма для введення даних про сорт культури зображена на рис.6.10.

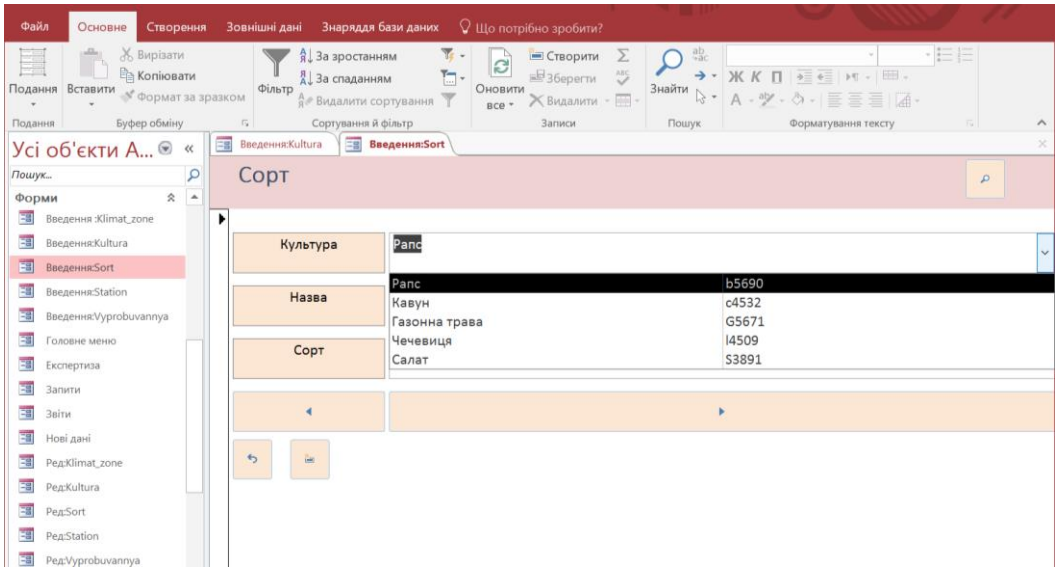


Рис.6.10 Форма «Сорт культури»

Створена форма «Кліматична зона» в режимі конструктора представлена на рис. 6.11.

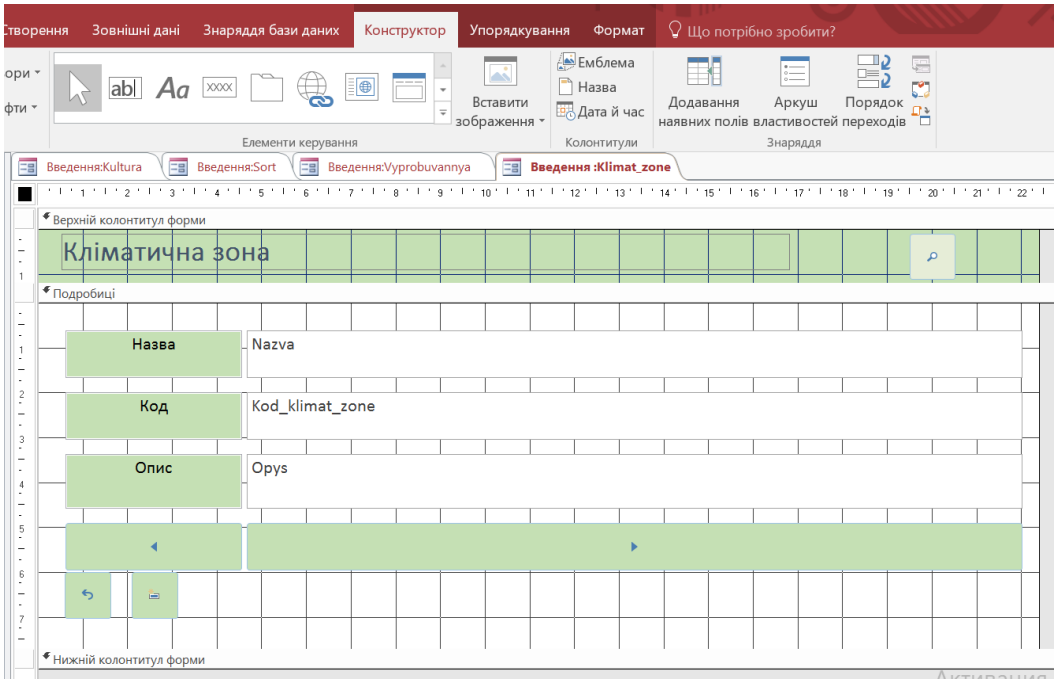


Рис.6.11 Форма «Кліматична зона» в режимі конструктора

Форма для введення даних про кліматичні зони зображена на рис.6.12.

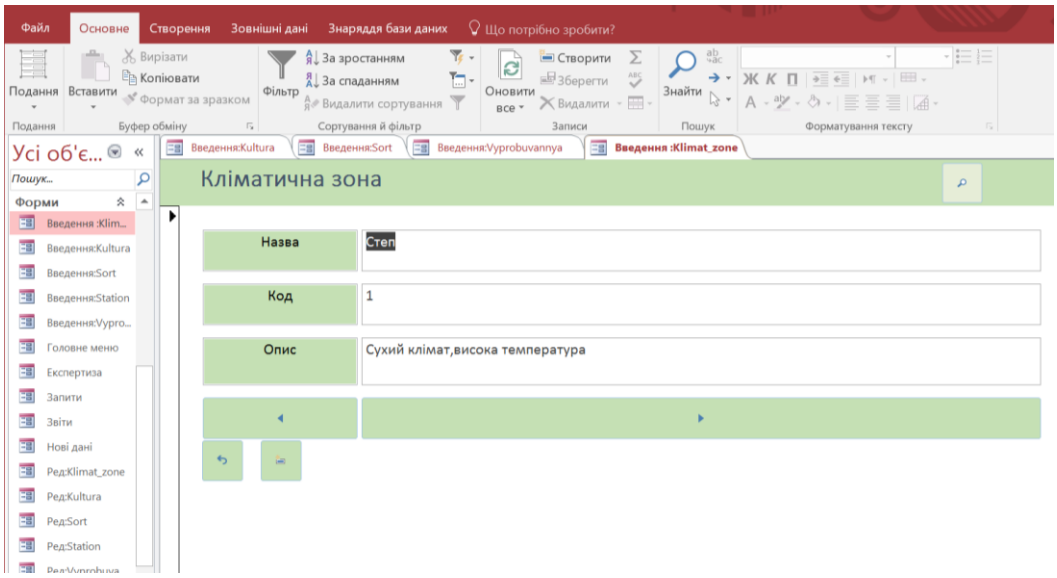


Рис.6.12 Форма «Кліматична зона»

Створена форма «Станція» в режимі конструктора представлена на рис. 6.13.

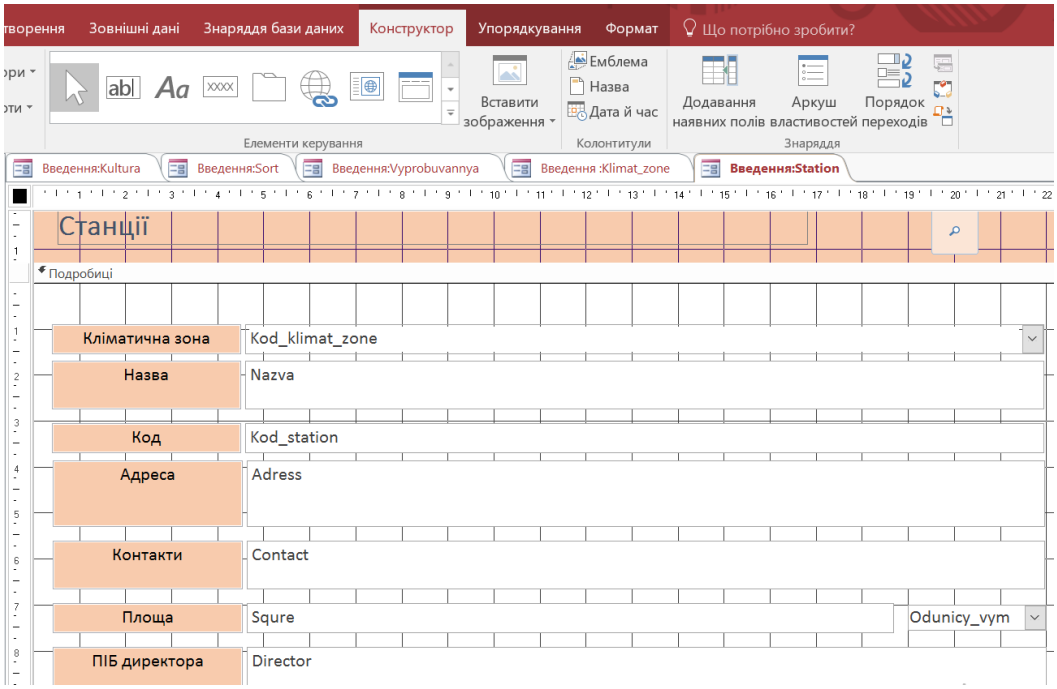


Рис.6.13 Форма «Станція» в режимі конструктора

Форма для введення даних про станції зображена на рис.6.14.

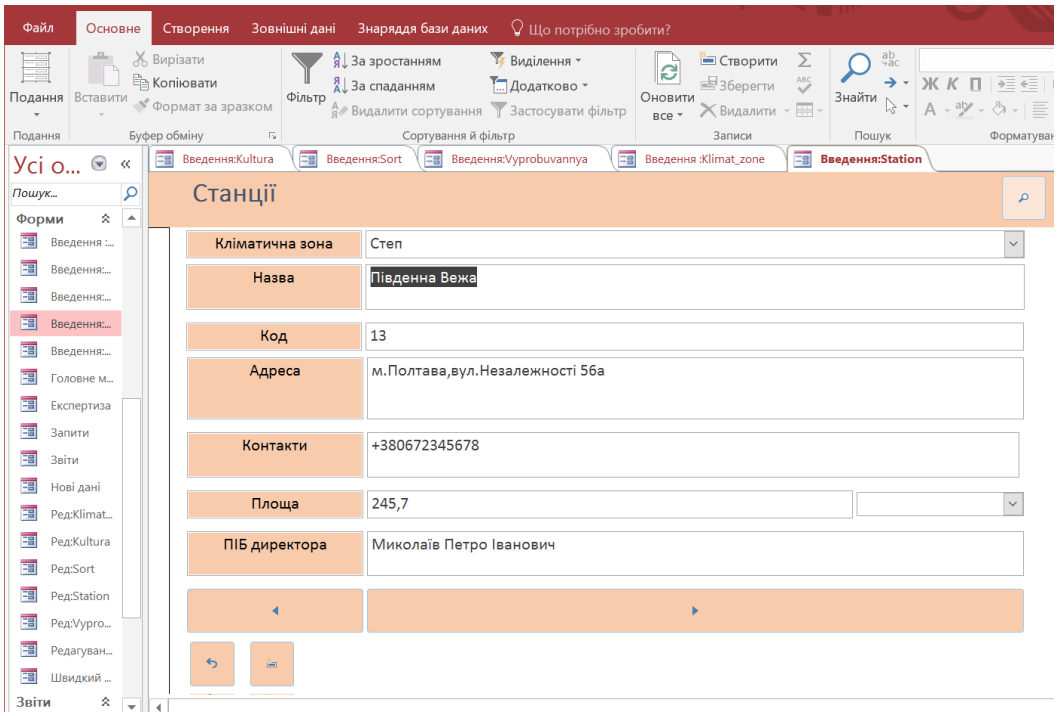


Рис.6.14 Форма «Станція»

Створена форма «Випробування» в режимі конструктора представлена на рис. 6.15.

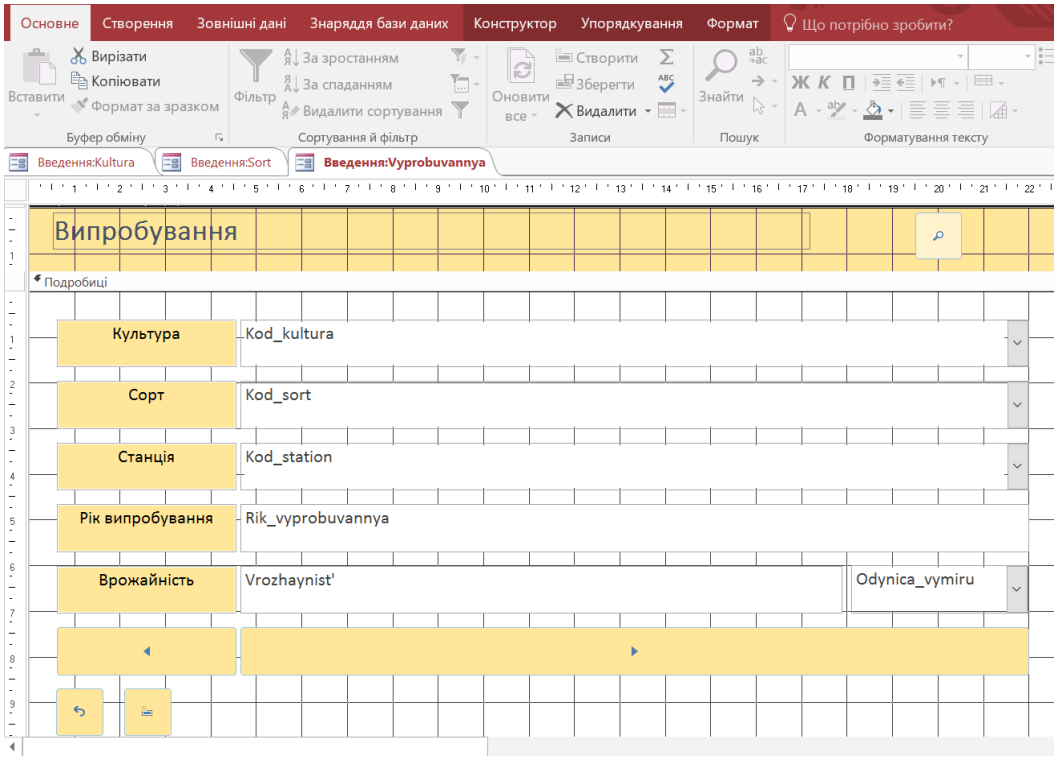


Рис.6.15 Форма «Випробування» в режимі конструктора

Форма для введення даних про випробування зображена на рис.6.16.

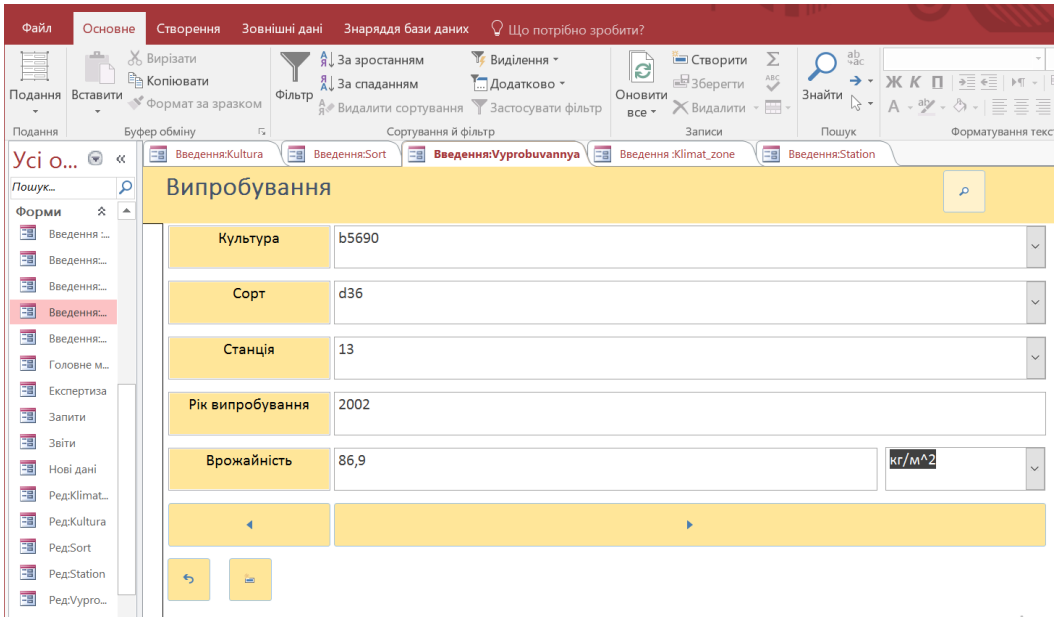


Рис.6.16 Форма «Випробування»

6.3 Створення запитів на вибірку

Теоретична частина

Запити використовуються для перегляду, зміни й аналізу даних різними способами. Запити також можна використовувати як джерела записів для форм, звітів і сторінок доступу до даних.

Найбільш розповсюджений тип запитів — запит на вибірку. Запит на вибірку відбирає дані з однієї чи більш таблиць за заданими умовами, а потім відображає їх у потрібному порядку.

Щоб спростити створення запитів, які будуть виконуватися або незалежно, або як базові для декількох форм чи звітів, можна скористатися майстром запитів. Майстер запитів автоматично виконує основні дії в залежності від відповідей користувача на поставлені питання. Якщо було створено кілька запитів, майстра можна також використовувати для швидкого створення структури запиту. Для корегування запиту необхідно переключитися у режим конструктора.

Постановка завдання

Основна мета лабораторної роботи – створення запитів на вибірку, які реалізують швидкий пошук тої чи іншої інформації за вимогою користувача.

Необхідно розробити запити, які відображають таку інформацію.

1. Список назв українською мовою усіх культур, занесених до бази даних. Список має бути представлений в алфавітному порядку.

2. Список усіх заявлених сортів вказаної користувачем культури. У списку вказати: назву культури українською мовою, назву сорту, код сорту. Представити список, упорядкований за назвою сорту в алфавітному порядку.

3. Список усіх кліматичних зон України. У списку вказати: назву, код та опис кліматичної зони. Вивести інформацію, упорядковану за кодом кліматичної зони.

4. Список дослідних станцій. У списку вказати: назву кліматичної зони, назву станції, адресу. Упорядкувати за назвою кліматичної зони та назвою станції.

5. Повну інформацію щодо дослідної станції, вказавши її назву.

6. Список усіх дослідних станцій, які розташовані у вказаній користувачем кліматичній зоні (назва) та на яких проходили випробування у вказаному користувачем році. У списку вказати: рік випробування, назву кліматичної зони, назву станції.

7. Список культур, які проходили випробування на станціях, що розташовані у вказаній кліматичній зоні. У списку вказати: назву кліматичної зони, назву станції, назву культури. Упорядкувати за усіма стовпцями.

8. Список сортів вказаної користувачем культури (код), які проходили випробування на вказаній користувачем станції (код). У списку вказати: назву культури, назву станції, назву сорту, рік випробування, врожайність. Упорядкувати за назвами та роком.

Завдання на виконання лабораторної роботи

1. Розробити запити відповідно постановці завдання.

2. У звіті представити скрін-шоти результатів виконання усіх запитів.

Приклад оформлення звітів

На рис. 6.17 зображено створення запиту в режимі конструктора список назв українською мовою, в алфавітному порядку усіх культур, занесених до бази даних.

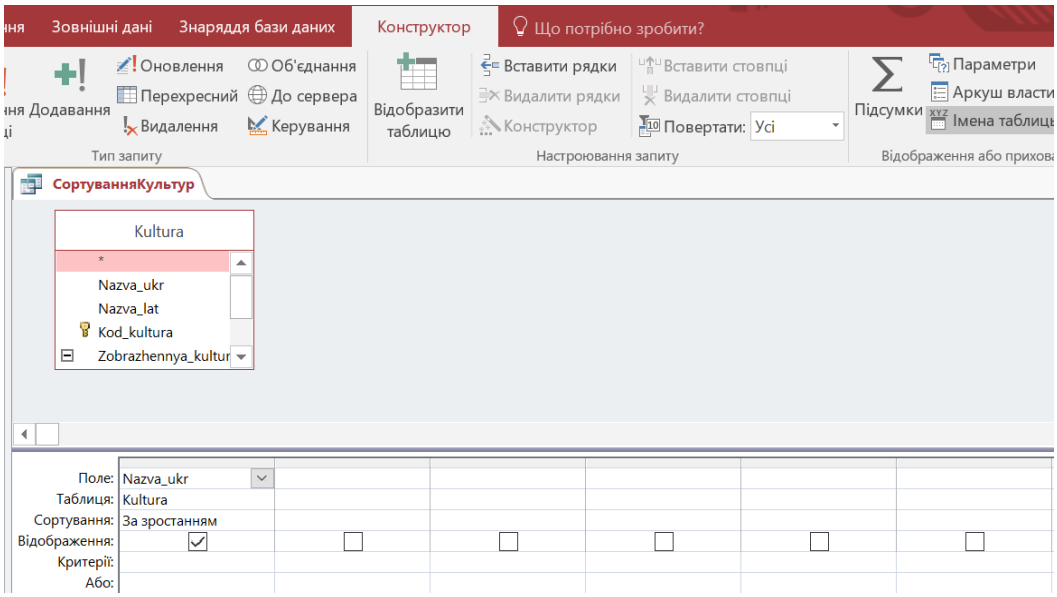


Рис.6.17 Конструктор запиту «Список назв українською мовою усіх культур»

На рис.6.18 зображено виконання запиту список назв українською мовою, в алфавітному порядку усіх культур, занесених до бази даних.

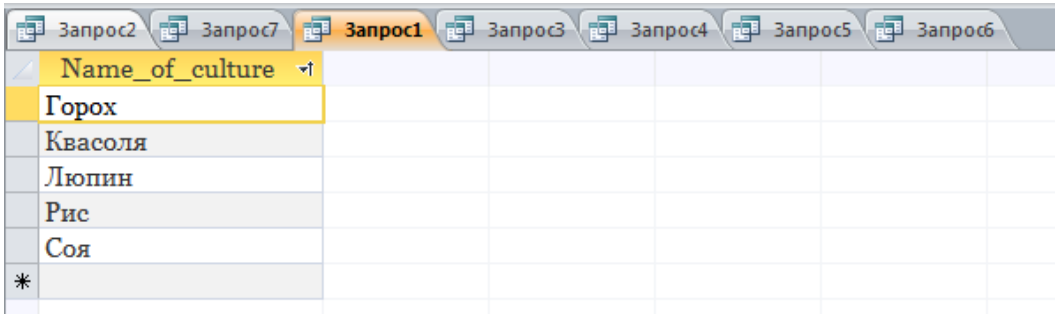


Рис.6.18 Список назв українською мовою усіх культур

На рис. 6.19 зображено створення запиту в режимі конструктора список усіх заявлених сортів культури вказаної культури.

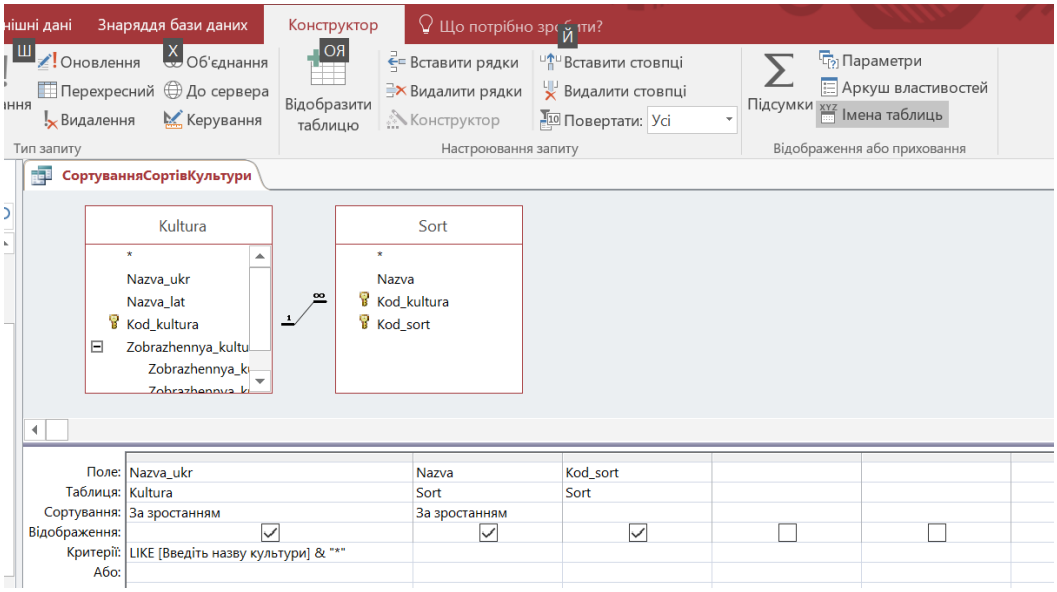


Рис.6.19 Конструктор запиту «Список усіх заявлених сортів культури»

На рис. 6.20 представлено виконання запиту список усіх заявлених сортів культури «Люпин».

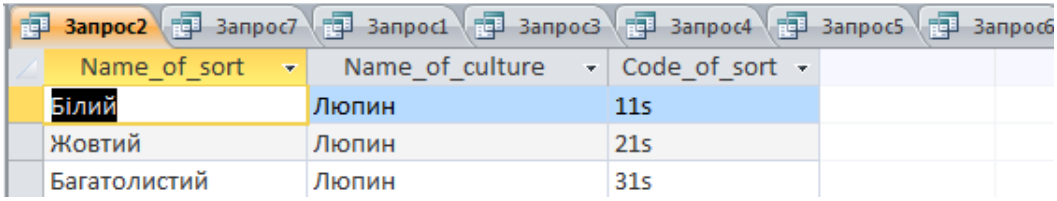


Рис.6.20 Список усіх заявлених сортів культури «Люпин»

На рис. 6.21 зображено створення запиту в режимі конструктора список усіх кліматичних зон України.

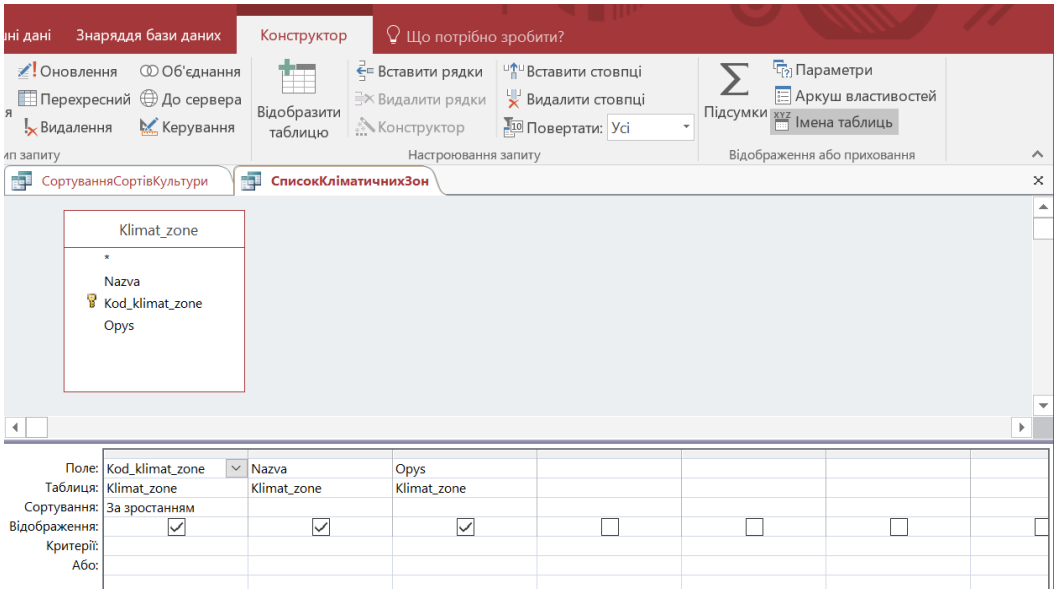


Рис.6.21 Конструктор запиту «Список усіх кліматичних зон України»

Виконання запиту список усіх кліматичних зон України зображено на рис.6.22.

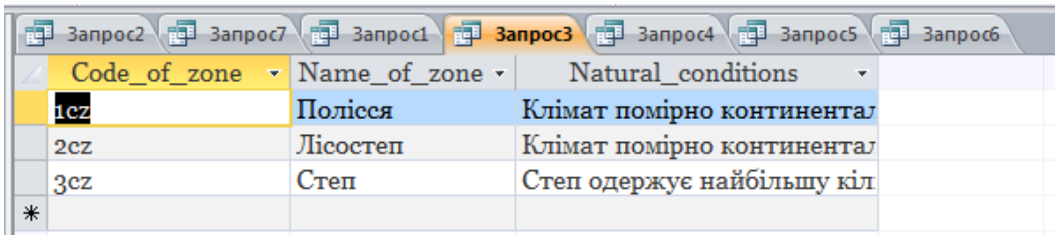


Рис.6.22 Список усіх кліматичних зон України

На рис. 6.23 зображено створення запиту в режимі конструктора список дослідних станцій, упорядкованих за назвою кліматичної зони та назвою станції.

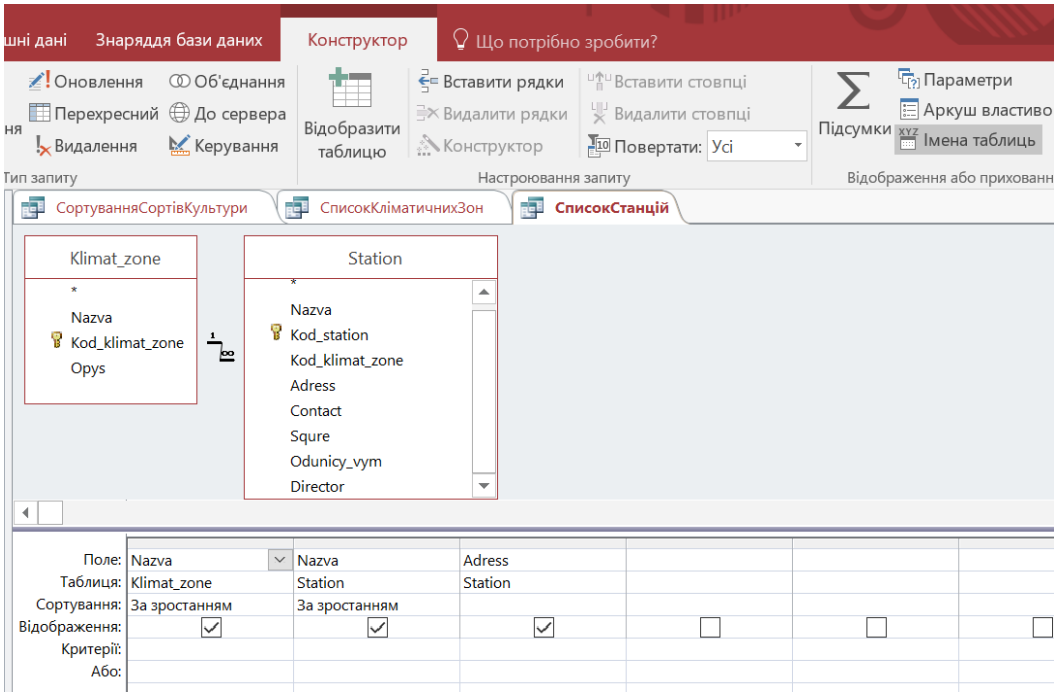
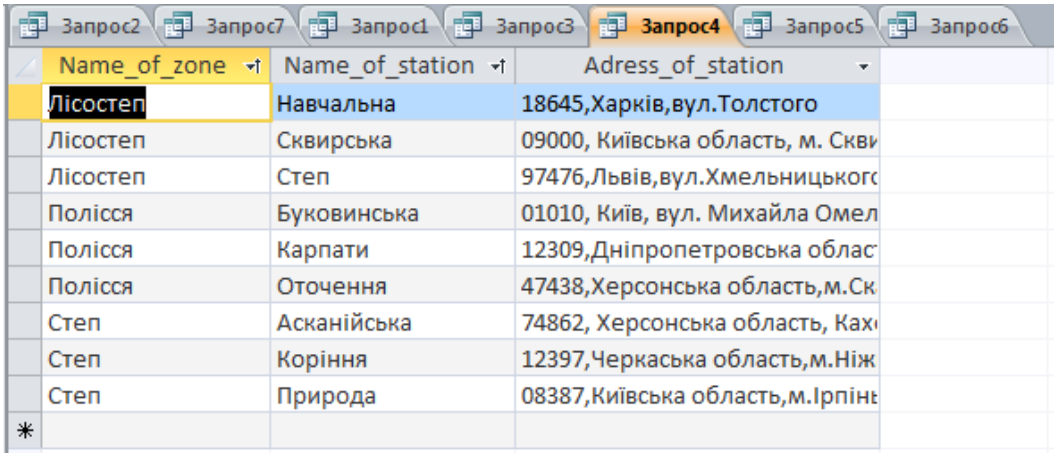


Рис.6.23 Конструктор запиту «Список дослідних станцій, упорядкованих за назвою кліматичної зони та назвою станції»

На рис. 6.24 подано виконання запиту список дослідних станцій, упорядкованих за назвою кліматичної зони та назвою станції.



Name_of_zone	Name_of_station	Adress_of_station
Лісостеп	Навчальна	18645,Харків,вул.Толстого
Лісостеп	Сквирська	09000, Київська область, м. Скви
Лісостеп	Степ	97476,Львів,вул.Хмельницького
Полісся	Буковинська	01010, Київ, вул. Михайла Омел
Полісся	Карпати	12309,Дніпропетровська облас
Полісся	Оточення	47438,Херсонська область,м.Ск
Степ	Асканійська	74862, Херсонська область, Кахо
Степ	Коріння	12397,Черкаська область,м.Ніж
Степ	Природа	08387,Київська область,м.Ірпінь

Рис.6.24 Список дослідних станцій

На рис. 6.25 зображено створення запиту в режимі конструктора повна інформація щодо вказаної дослідної станції.

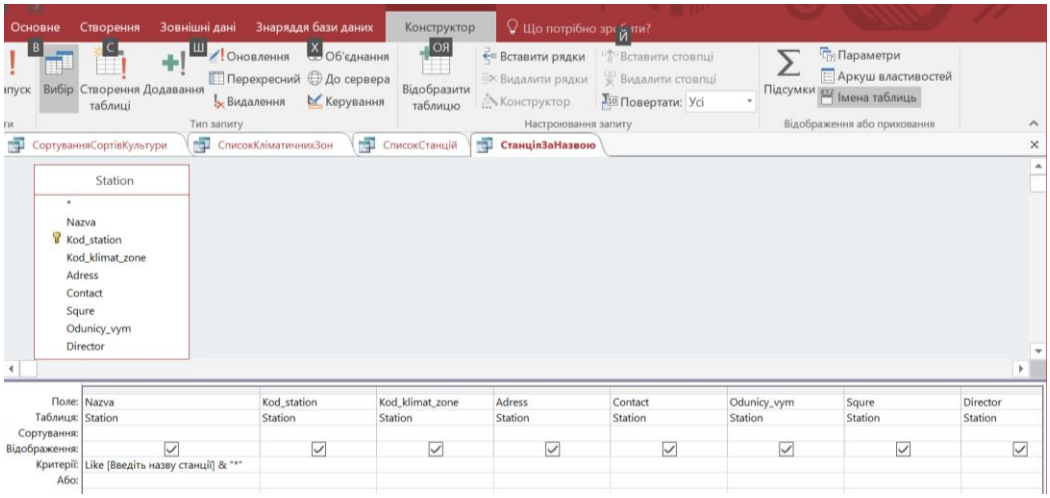


Рис.6.26 Конструктор запиту «Повна інформація щодо вказаної дослідної станції»

Повну інформацію щодо дослідної станції «Асканійська» зображено на рис. 6.27.

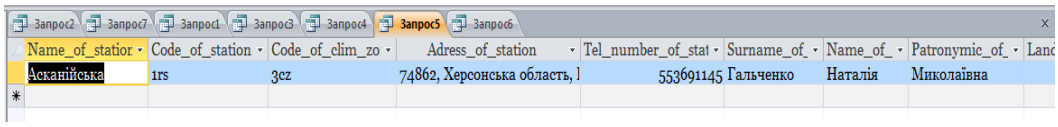


Рис.6.27 Повна інформація щодо дослідної станції «Асканійська»

На рис. 6.28 зображено створення запиту в режимі конструктора список усіх дослідних станцій, які розташовані у вказаній кліматичній зоні та у вказаному році.

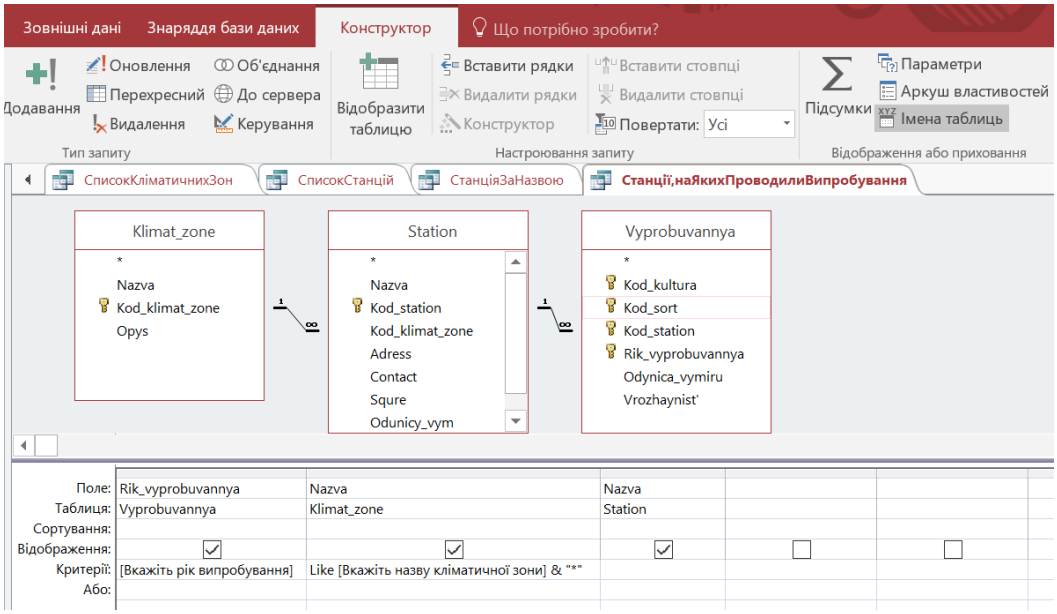


Рис.6.28 Конструктор запиту «Список усіх дослідних станцій, які розташовані у вказаній кліматичній зоні та у вказаному році»

На рис.6.29 відображено список усіх дослідних станцій, які розташовані у вказаній кліматичній зоні «Степ», та на яких проходили випробування у 2017 році.

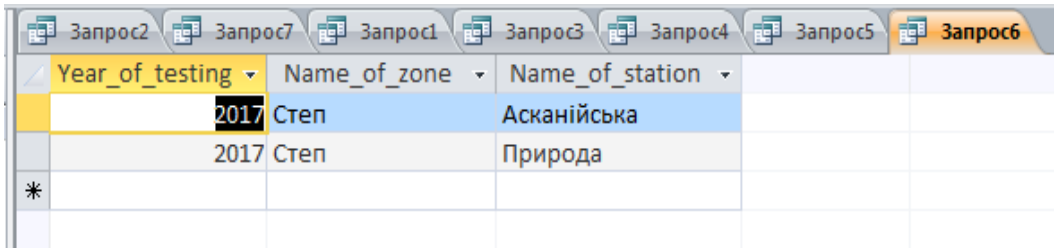


Рис.6.29 Список дослідних станцій у кліматичній зоні «Степ»

На рис. 6.30 зображено створення запиту в режимі конструктора список культур, які проходили випробування на станціях у вказаній кліматичній зоні.

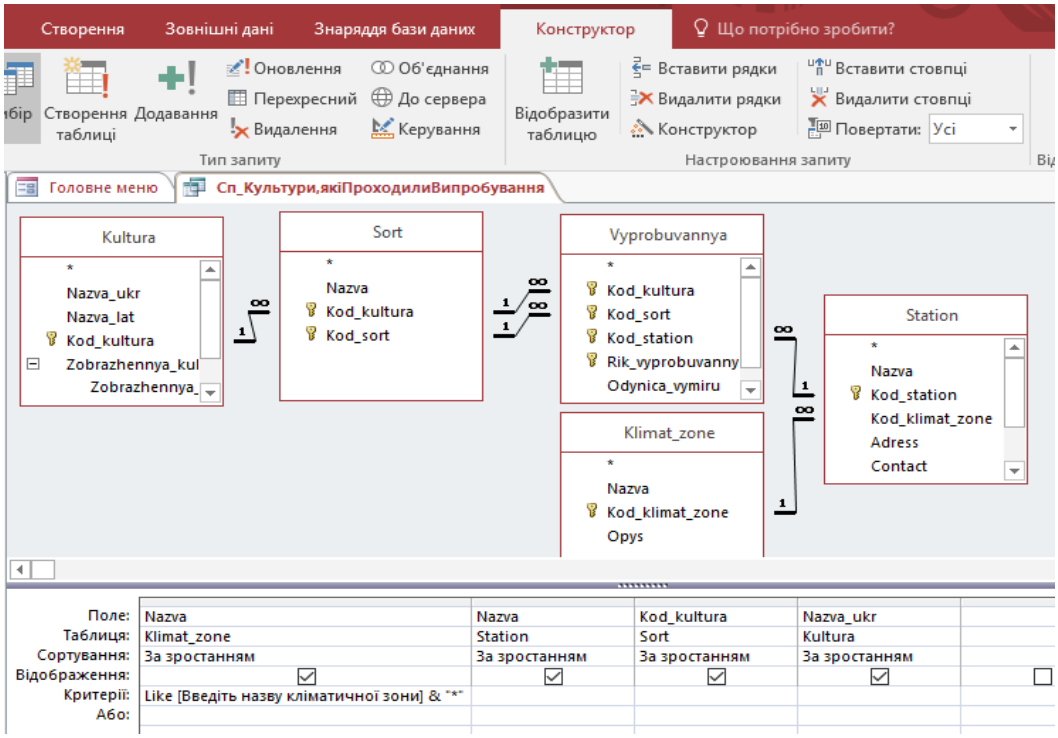
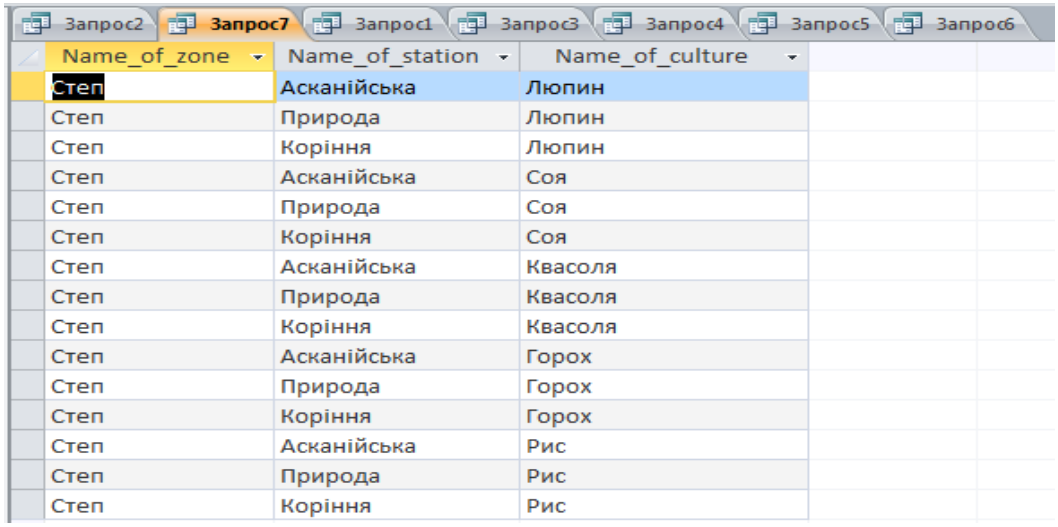


Рис.6.30 Конструктор запиту «Список культур, які проходили випробування на станціях у вказаній кліматичній зоні»

На рис. 6.31 зображено список культур, які проходили випробування на станціях у кліматичній зоні «Степ».



Name_of_zone	Name_of_station	Name_of_culture
Степ	Асканійська	Люпин
Степ	Природа	Люпин
Степ	Коріння	Люпин
Степ	Асканійська	Соя
Степ	Природа	Соя
Степ	Коріння	Соя
Степ	Асканійська	Квасоля
Степ	Природа	Квасоля
Степ	Коріння	Квасоля
Степ	Асканійська	Горох
Степ	Природа	Горох
Степ	Коріння	Горох
Степ	Асканійська	Рис
Степ	Природа	Рис
Степ	Коріння	Рис

Рис. 6.31 Список культур, які проходили випробування на станціях у кліматичній зоні «Степ»

6.4 Створення звітів

Теоретична частина

Звіт являє собою ефективний засіб представлення даних в друкованому вигляді. Маючи можливість керувати розміром та зовнішнім виглядом всіх елементів звіту, користувач може відображувати інформацію у бажаному вигляді.

Більша частина інформації у звіті надається із таблиці бази даних, запиту або інструкції SQL, які є джерелом даних для звіту. Інша інформація звіту зберігається у його структурі.

Для створення зв'язку між звітом та його джерельними даними використовуються елементи управління (поля, надписи, декоративні лінії для оформлення звіту тощо).

Постановка завдання

Основна мета лабораторної роботи – створення документів, які формують списки відповідно розробленим запитам у лабораторній роботі №3. Кожний документ має мати назву, під якою вказується дата його формування.

Таким чином, необхідно розробити такі звіти.

1. Назва звіту «Список культур, занесених до бази даних, на …(*дата)*». Шапка містить два стовпчики: «№ з/п» та «Назва культури».

2. Назва звіту: «Список усіх заявлених сортів … (*назва культури),* , на …(*дата)»*. Шапка містить стовпчики: «№ з/п», «Назва сорту», «Код сорту».

3. Назва звіту: «Список усіх кліматичних зон України». Шапка містить стовпчики: «№ з/п», «Назва кліматичної зони України», «Код кліматичної зони», «Характеристика».

4. Назва звіту: «Список дослідних станцій України». Шапка містить стовпчики: «Назва кліматичної зони» (повторюється один раз), «Назва станції», «Адреса».

5. Назва звіту: «Список усіх дослідних станцій, які розташовані у … (*назва кліматичної зони*), на яких проходили випробування у … (*рік*)». Шапка містить стовпчики: «№ з/п», «Назва станції».

6. Назва звіту: «Список культур, які проходили випробування на станціях, що розташовані у … (*назва кліматичної зони*)». Шапка містить стовпчики: «Назва станції» (повторюється один раз), «Назва культури».

7. Назва звіту: «Список сортів … (*назва культури*), які проходили випробування на … (*назва станції*)». Шапка містить стовпчики: «Назва сорту», «Рік випробування», «Врожайність».

Завдання на виконання лабораторної роботи

1. Розробити вказані списки відповідно постановці завдання.
2. У звіті за лабораторною роботою представити усі звіти.

Приклад оформлення звітів

Звіт «Список культур» подано на рис. 6.32.

Список культур, занесених до бази даних, на 10 листопада 2017 р.

	Назва
1.	Газонна трава
2.	Кавун
3.	Рапс
4.	Салат
5.	Чечевиця

Сторінка 1 з 1

Рис.6.32 Звіт «Список культур»

Список усіх заявлених сортів за культурою «Рапс» (рис.6.33).

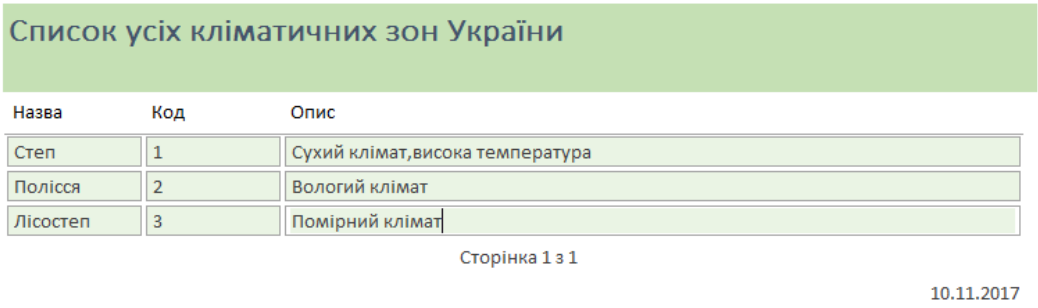
Список усіх заявлених сортів культури Рапс на 10 листопада 2017 р.

	Назва	Код
1.	Джером	d36
2.	Нельсон	n33
3.	Сальса	s34

Сторінка 1 з 1

Рис.6.33 Список усіх заявлених сортів за культурою «Рапс»

Звіт «Список усіх кліматичних зон України» (рис.6.34).



Список усіх кліматичних зон України

Назва	Код	Опис
Степ	1	Сухий клімат,висока температура
Полісся	2	Вологий клімат
Лісостеп	3	Помірний клімат

Сторінка 1 з 1

10.11.2017

Рис.6.34 Звіт «Список усіх кліматичних зон України»

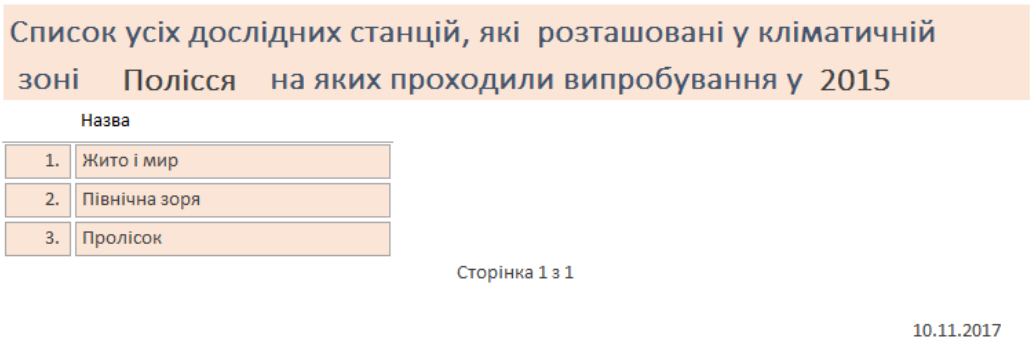
На рис.6.35 звіт під назвою «Список станцій» (список дослідних станцій України).

Список дослідних станцій України

Кліматична зона	Назва	Адреса
Лісостеп		
	Дружба	Київська обл.,м.Вишневе,вул.Соборська 121в
	Зелена зона	Сумська обл.,с.Кровне,вул.Рупіних 67
	Соняшникова	Сумська обл.,с.Ямне,вул.Яблунева 45
Полісся		
	Жито і мир	м.Житомир,вул.Караваївська 54
	Північна зоря	м.Івано-Франківськ,вул.Львова 78/9
	Пролісок	м.Ужгород,вул.Бандерівська 56
Степ		
	Луг	м.Миколаїв,вул.Слобідська 67
	Огник	м.Кременчуг,вул.Соборна 36
	Південна Вежа	м.Полтава,вул.Незалежності 56а

Рис.6.35 Звіт «Список станцій»

На рис.6.36 зображено звіт «Список станцій, які розташовані у кліматичній зоні Полісся, на яких проходили випробування у 2015 році».



Список усіх дослідних станцій, які розташовані у кліматичній зоні Полісся на яких проходили випробування у 2015

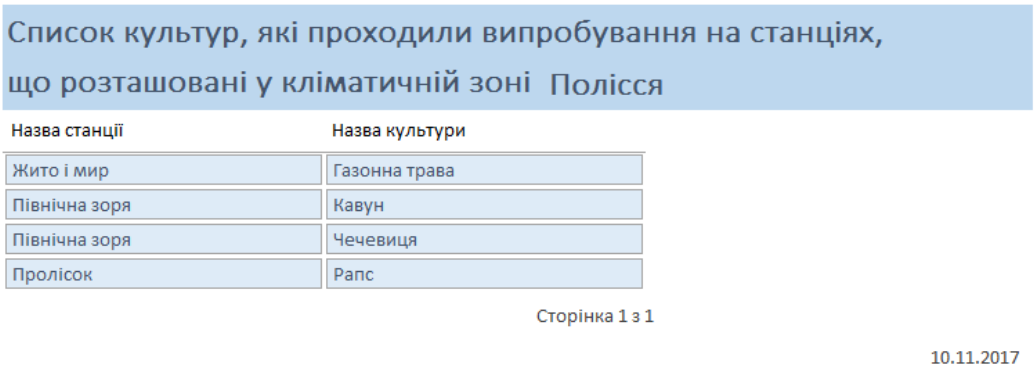
	Назва
1.	Жито і мир
2.	Північна зоря
3.	Пролісок

Сторінка 1 з 1

10.11.2017

Рис.6.36 Звіт «Список станцій за кліматичною зоною і роком»

На рис.6.37 зображено звіт «Список культур, які проходили на станція, що розташовані у кліматичній зоні Полісся».



Список культур, які проходили випробування на станціях, що розташовані у кліматичній зоні Полісся

Назва станції	Назва культури
Жито і мир	Газонна трава
Північна зоря	Кавун
Північна зоря	Чечевиця
Пролісок	Рапс

Сторінка 1 з 1

10.11.2017

Рис.6.37 Звіт «Список культур за зоною»

На рис.6.38 відображений звіт «Список сортів за вказаною культурою та станцією».

Список сортів культури Рапс які проходили випробування на станції Пролісок

Назва	Рік випробування	Врожайність	
Джером			
	2015	82,9	кг/м^2
	2004	73,4	кг/м^2
	2003	95,8	кг/м^2
	2002	52,9	кг/м^2
	2001	13,4	кг/м^2

Сторінка 1 з 1

10.11.2017

Рис.6.39 Звіт «Список сортів за культурою та станцією»

6.5 Створення кнопкової форми

Теоретична частина

Кнопкова форма – це засіб, що дозволяє розробити інтерфейс користувача у вигляді діалогового вікна, який реалізує навігацію по різним функціям системи (проекту).

Кнопкова форма складається лише з кнопок і написів. Вона не прив’язується ні до таблиць, ні до запитів. Але, в решті решт, натискуючи на ту чи іншу кнопку, користувач має можливість увійти у

форму, що дозволить йому змінювати інформацію у базі даних, реалізовувати пошук та формувати звіти.

Самі кнопкові форми розроблюються у режимі конструктора. Викладаючи на форму кнопку, розробник має можливість покроково вже у режимі майстра вказати призначення кнопки та змінити її вигляд.

Постановка завдання

Основна мета лабораторної роботи – розробка інтерфейсу користувача для керування системи в цілому.

При завантаженні MS Access користувач має побачити головне вікно, яке дає йому можливість вибрати ту чи іншу операцію з даними. На рис. 6.40 зображено можливий вигляд цього вікна.

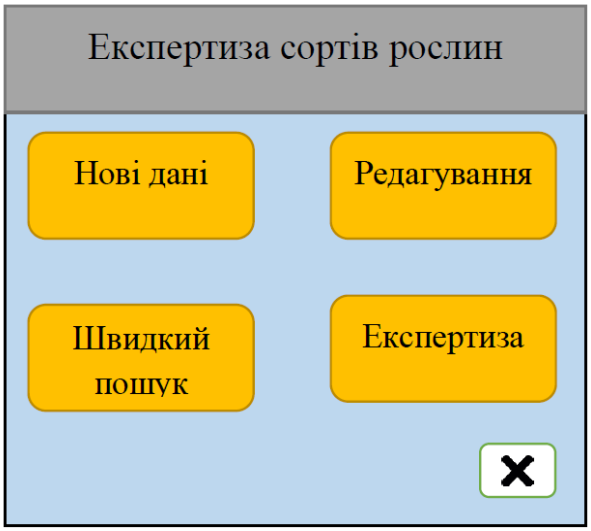


Рис.6.40 Головне вікно

При натисканні на окрему кнопку має зв’явитися наступне вікно. Наприклад, якщо натиснути на кнопку «Нові дані», користувачеві надається можливість вносити нову інформацію у базу даних. Вікно, до якого він переходить, може бути таким, як на рис. 6.41.

Рис. 6.41 Внесення нових даних

При цьому при натисканні на будь-яку кнопку у вікні «Нові дані» має з’явитися відповідна форма для внесення даних, одна з тих, що розроблювалися у лабораторній роботі №2. Необхідно перевірити, що у властивостях форми зазначено лише режим додавання записів.

Якщо ж вибрана операція «Редагування», користувач має побачити майже таке саме вікно, що і у режимі «Нові дані», але з іншою назвою та з іншими властивостями форми.

Якщо ж у головному вікні вибрати операцію «Швидкий пошук», то має з’явитися вікно, як на рис. 6.42.

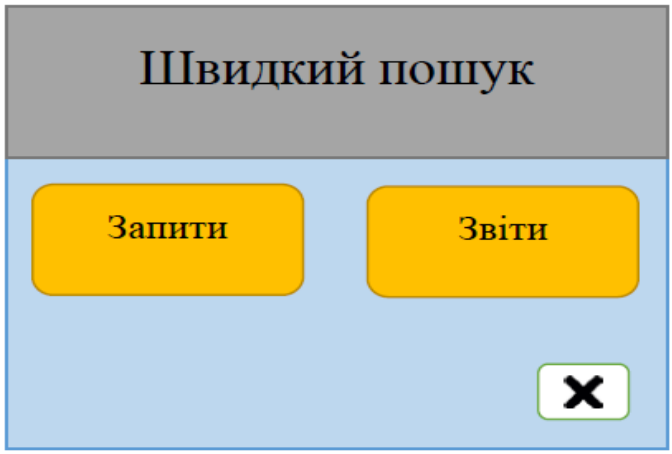


Рис. 6.42 Опції швидкого пошуку

При натисканні на кнопку «Запити» має з’явитися вікно з можливістю виконати будь-який запит з лабораторної роботи №3, а при натисканні на кнопку «Звіти» - будь-який звіт з лабораторної роботи №4.

Якщо ж у головному вікні натиснути на кнопку «Експертиза», то має з’явитися можливість виконання певних алгоритмів обробки даних, які і реалізують основну мету проекту – отримання висновку щодо можливості розповсюдження на території України заявлених сортів рослин. Ці алгоритми ми будемо розроблювати у наступних лабораторних роботах.

Завдання на виконання лабораторної роботи

1. Розробити форми і реалізувати усі можливості навігації вікон «Експертиза сортів рослин», «Нові дані», «Редагування», «Швидкий пошук», «Запити», «Звіти».

2. У звіті за лабораторною роботою представити відповідні скрін-шоти.

Приклад оформлення звітів

Головна кнопкова форма зображена на рис.6.43.

Рис.6.43 Головна кнопкова форма

Після натиснення на головній формі на кнопку «Нові дані» з’являється форма «Нові дані» зображена на рис.6.44.

Рис.6.44 Форма «Нові дані»

Після натиснення на формі «Нові дані» на кнопку «Культури» з’являється форма додавання нових даних про культуру з двома кнопкою дадати запис (рис.6.45).

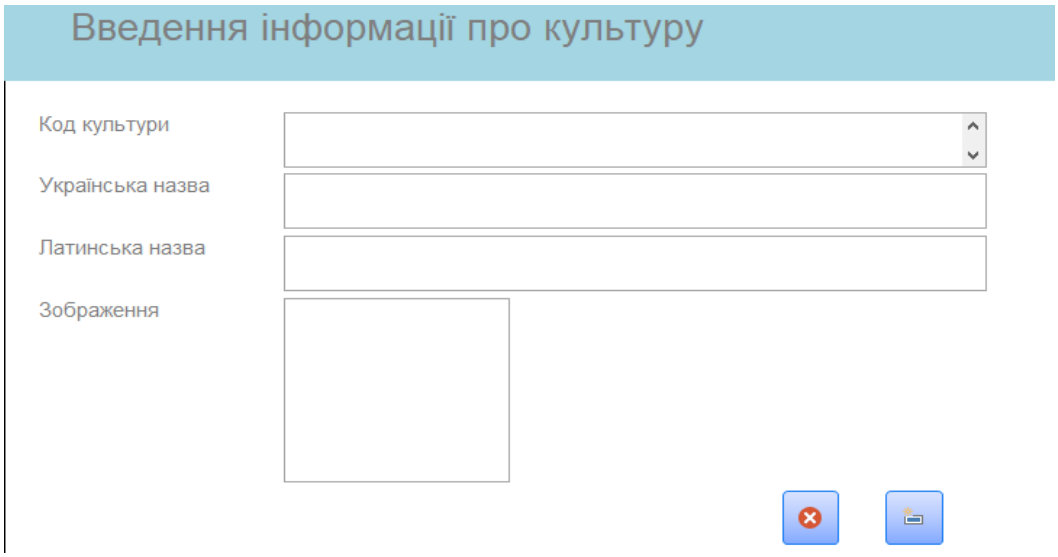


Рис.6.45 Форма додавання нових даних про культуру

Повернувшись до головної форми за допомогою кнопки «До попереднього вікна» (рис.6.43) натискаємо на формі (рис.6.46) на кнопку «Редагування» та з’являється форма «Редагування» зображена на рис.6.46.

Рис.6.46 Форма «Редагування»

Після натиснення на формі «Редагування» на кнопку «Сорти» з’являється форма редагування існуючиз даних про сорти культури з двома кнопкою видалити запис (рис.6.47).

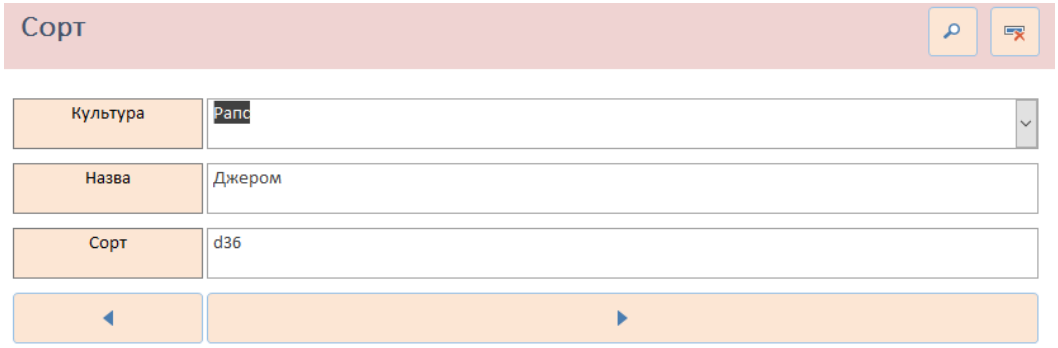


Рис.6.47 Форма редагування даних про сорти культури

Повернувшись до головної форми за допомогою кнопки «До попереднього вікна» (рис.6.43) натискаємо на формі кнопку «Швидкий пошук» та з’являється відповідна форма, на якій розміщенні кнопки переходу «Запити» та «Звіти» зображена на рис.6.48.

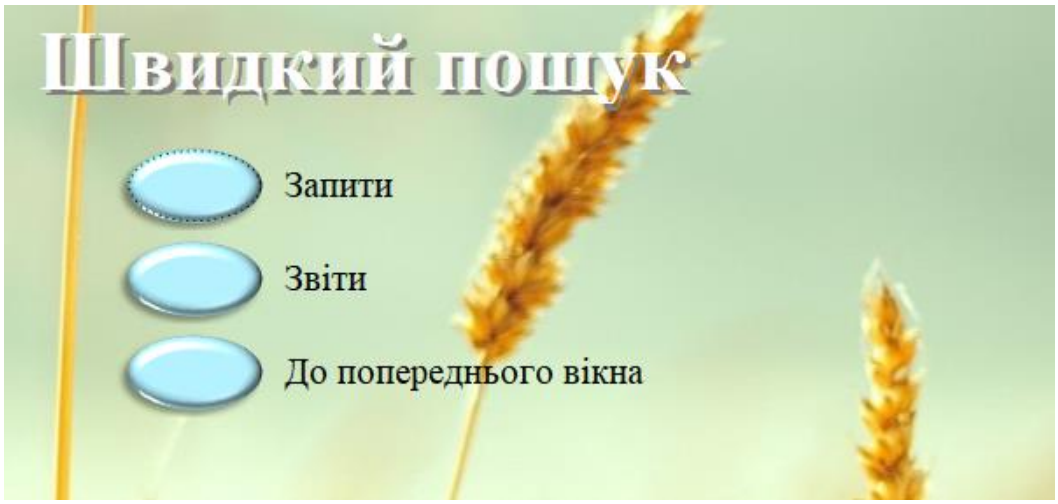


Рис.6.48 Форма «Швидкий пошук»

Після натиснення на формі «Швидкий пошук» на кнопку «Запити» з’являється форма з кнопками виконання запитів, які були створені в 6.3 (рис.6.49).

Рис.6.49 Форма «Запити»

Після натиснення на формі «Швидкий пошук» на кнопку «Звіти» з’являється форма з кнопками виконання звітів, які були створені в 6.4 (рис.6.50).

Рис.6.50 Форма «Звіти»

6.6 Створення запитів на вибірку мовою SQL

Постановка завдання

Основна мета – створення запитів на вибірку, які реалізують швидкий пошук тої чи іншої інформації за вимогою користувача (лабораторна робота №3) мовою SQL.

Необхідно розробити запити, які відображають таку інформацію.

1. Список назв українською мовою усіх культур, занесених до бази даних. Список має бути представлений в алфавітному порядку.

2. Список усіх заявлених сортів вказаної користувачем культури. У списку вказати: назву культури українською мовою, назву сорту, код сорту. Представити список, упорядкований за назвою сорту в алфавітному порядку.

3. Список усіх кліматичних зон України. У списку вказати: назву, код та опис кліматичної зони. Вивести інформацію, упорядковану за кодом кліматичної зони.

4. Список дослідних станцій. У списку вказати: назву кліматичної зони, назву станції, адресу. Упорядкувати за назвою кліматичної зони та назвою станції.

5. Повну інформацію щодо дослідної станції, вказавши її назву.

6. Список усіх дослідних станцій, які розташовані у вказаній користувачем кліматичній зоні (назва) та на яких проходили випробування у вказаному користувачем році. У списку вказати: рік випробування, назву кліматичної зони, назву станції.

7. Список культур, які проходили випробування на станціях, що розташовані у вказаній кліматичній зоні. У списку вказати: назву кліматичної зони, назву станції, назву культури. Упорядкувати за усіма стовпцями.

8. Список сортів вказаної користувачем культури (код), які проходили випробування на вказаній користувачем станції (код). У списку вказати: назву культури, назву станції, назву сорту, рік випробування, врожайність. Упорядкувати за назвами та роком.

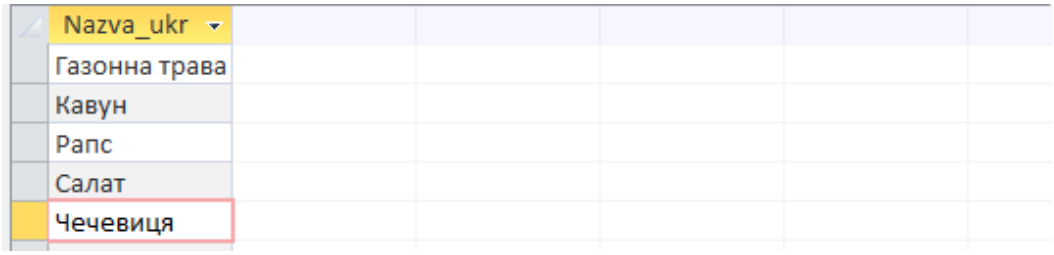
Приклад оформлення звітів

Запит «Список назв українською мовою усіх культур, занесених до бази даних. Список має бути представлений в алфавітному порядку» представлено на рис.6.51.

```
SELECT Kultura.Nazva_ukr
FROM Kultura
ORDER BY Kultura.Nazva_ukr;
```

Рис.6.51 Запит на вибірку списку культур

Виконання запиту представлено на рис.6.52.



Nazva_ukr
Газонна трава
Кавун
Рапс
Салат
Чечевиця

Рис.6.53 Список назв українською мовою усіх культур

Запит «Список усіх заявлених сортів вказаної користувачем культури. У списку вказати: назву культури українською мовою, назву сорту, код сорту. Представити список, упорядкований за назвою сорту в алфавітному порядку» зображено на рис.6.54.

```
SELECT Kultura.Nazva_ukr, Sort.Nazva, Sort.Kod_sort
FROM Kultura INNER JOIN Sort ON Kultura.Kod_kultura =
Sort.Kod_kultura
WHERE (((Kultura.Nazva_ukr) like [Введіть назву
культури]&"*"))
ORDER BY Kultura.Nazva_ukr, Sort.Nazva;
```

Рис.6.54 Запит на вибірку списку сортів вказаної культури

Запит «Список усіх кліматичних зон України. У списку вказати: назву, код та опис кліматичної зони. Вивести інформацію, упорядковану за кодом кліматичної зони» зображено на рис.6.55.

```
SELECT Klimat_zone.Kod_klimat_zone, Klimat_zone.Nazva,
Klimat_zone.Opys
FROM Klimat_zone
ORDER BY Klimat_zone.Kod_klimat_zone;
```

Рис.6.55. Запит на вибірку списку кліматичних зон України

Запит «Список дослідних станцій. У списку вказати: назву кліматичної зони, назву станції, адресу. Упорядкувати за назвою кліматичної зони та назвою станції» зображено на рис.6.56.

```
SELECT Klimat_zone.Nazva, Station.Nazva, Station.Adress
FROM Klimat_zone INNER JOIN Station ON
Klimat_zone.Kod_klimat_zone = Station.Kod_klimat_zone
ORDER BY Klimat_zone.Nazva, Station.Nazva;
```

Рис.6.56 Запит на вибірку списку дослідних станцій

Запит «Повну інформацію щодо дослідної станції, вказавши її назву» зображено на рис.6.57.

```
SELECT Station.Nazva, Station.Kod_station,
Station.Kod_klimat_zone, Station.Adress, Station.Contact,
Station.Odunicy_vym, Station.Squre, Station.Director
FROM Station
WHERE (((Station.Nazva) like [Введіть назву станції]&"*"));
```

Рис.6.57 Запит на вибірку заданої дослідної станції

Запит «Список усіх дослідних станцій, які розташовані у вказаній користувачем кліматичній зоні (назва) та на яких проходили випробування у вказаному користувачем році. У списку вказати: рік випробування, назву кліматичної зони, назву станції» зображено на рис.6.58.

```
SELECT DISTINCT Vyprobuvannya.Rik_vyprobuvannya,
Klimat_zone.Nazva, Station.Nazva
FROM (Klimat_zone INNER JOIN Station ON
Klimat_zone.Kod_klimat_zone = Station.Kod_klimat_zone)
INNER JOIN Vyprobuvannya ON Station.Kod_station =
Vyprobuvannya.Kod_station
WHERE (((Vyprobuvannya.Rik_vyprobuvannya)=[Вкажіть рік
випробування]) And ((Klimat_zone.Nazva) Like [Вкажіть назву
кліматичної зони]&"*"));
```

Рис.6.58 Запит на вибірку списку дослідних станцій у вказаній кліматичній зоні, на яких проходили випробування за певний рік

Запит «Список культур, які проходили випробування на станціях, що розташовані у вказаній кліматичній зоні. У списку вказати: назву кліматичної зони, назву станції, назву культури. Упорядкувати за усіма стовпцями» зображено на рис.6.59.

```
SELECT DISTINCT Klimat_zone.Nazva, Station.Nazva,
Sort.Kod_kultura, Kultura.Nazva_ukr
FROM (Klimat_zone INNER JOIN Station ON
Klimat_zone.Kod_klimat_zone = Station.Kod_klimat_zone)
INNER JOIN ((Kultura INNER JOIN Sort ON
Kultura.Kod_kultura = Sort.Kod_kultura) INNER JOIN
Vyprobuvannya ON (Sort.Kod_kultura =
Vyprobuvannya.Kod_kultura) AND (Sort.Kod_sort =
Vyprobuvannya.Kod_sort)) ON Station.Kod_station =
Vyprobuvannya.Kod_station
WHERE (((Klimat_zone.Nazva) like [Введіть назву кліматичної
зони]&"*"))
ORDER BY Klimat_zone.Nazva, Station.Nazva, Sort.Kod_kultura,
Kultura.Nazva_ukr;
```

Рис.6.59 Запит на вибірку списку культур, які проходили випробування на станціях, що розташовані у вказаній кліматичній зоні

Запит «Список сортів вказаної користувачем культури (код), які проходили випробування на вказаній користувачем станції (код). У списку вказати: назву культури, назву станції, назву сорту, рік випробування, врожайність. Упорядкувати за назвами та роком» зображено на рис.6.60.

```
SELECT Kultura.Nazva_ukr, Sort.Nazva, Station.Nazva,
Vyprobuvannya.Rik_vyprobuvannya,
Vyprobuvannya.[Vrozhaynist'], Vyprobuvannya.Odynica_vymiru
FROM Station INNER JOIN ((Kultura INNER JOIN Sort ON
Kultura.Kod_kultura = Sort.Kod_kultura) INNER JOIN
Vyprobuvannya ON (Sort.Kod_kultura =
Vyprobuvannya.Kod_kultura) AND (Sort.Kod_sort =
Vyprobuvannya.Kod_sort)) ON Station.Kod_station =
Vyprobuvannya.Kod_station
WHERE (((Vyprobuvannya.Kod_kultura) LIKE [Введіть код
культури]&"*") AND ((Vyprobuvannya.Kod_station)=[Введіть
код станції]))
ORDER BY Kultura.Nazva_ukr, Sort.Nazva, Station.Nazva,
Vyprobuvannya.Rik_vyprobuvannya;
```

Рис.6.60 Запит на вибірку списку сортів вказаної культури

6.7 Експертна оцінка нових сортів рослин

Теоретична частина

Головна мета проекту, що розроблюється, визначити можливість розповсюдження нових сортів рослин і внесення нового сорту до реєстру сортів рослин України. Після збору врожаю остаточно визначаються усі необхідні параметри польових випробувань, головний з яких – врожайність.

Отримані параметри досліджуються усередині певної кліматичної зони. Як правило, новий сорт може показати хорошу врожайність лише в одній з кліматичних зон, хоча бувають і такі «рекордсмени», що показують хорошу врожайність в усіх трьох кліматичних зонах. Такі «рекордсмени» - мрія будь-якого замовника нового сорту.

Таким чином, для вирішення питань, пов’язаних з експертизою нового сорту, необхідно провести аналіз даних, що занесені до бази даних. Для цього розроблюються запити, на основі яких формуються звіти.

Постановка завдання

Основна мета лабораторної роботи – розробка звітів, за якими робиться висновок щодо можливості розповсюдження нового сорту рослини на території України та внесення такого сорту до реєстру сортів рослин України. Усі звіти будуть ґрунтуватися на запитах, які мають бути розроблені мовою SQL.

У лабораторній роботі №5 було представлено головне вікно проекту, в якому передбачена кнопка «Експертиза». При натисканні на цю кнопку користувач має побачити вікно з переліком звітів, які необхідні для проведення експертизи. Натискаючи на назву звіту, на екрані мають з’явитися необхідні діалогові вікна та сам звіт.

Завдання на виконання лабораторної роботи

1. Розробити форму «Експертні звіти» та представити у її осередку у вигляді кнопок усі відповідні звіти.

2. Розробити запити мовою SQL за таким завданням:

1) користувач має вказати культуру, кліматичну зону, станцію, часовий період випробувань (роки); на основі цих даних вивести усі сорти вказаної культури, вказавши їхню середню врожайність за вказаний період, упорядковуючи за врожайністю; передбачити варіанти:

а) за усіма станціям вказаної кліматичної зони;

б) за один рік;

в) лише один сорт та комбінації цих варіантів.

2) користувач вказує назву культури, кліматичну зону, мінімальну та максимальну значення врожайності; на основі цих даних вивести лише ті сорти вказаної культури, які у поточному році завершили

випробування та мають середню врожайність, що знаходиться у вказаному діапазоні врожайності;

3) користувач вказує назву культури та часовий період випробування; на основі цих даних вивести усі сорти вказаної культури, вказавши їхню середню врожайність, об'єднавши їх в окремі групи відповідно кліматичних зонах;

4) користувач вказує назву культури; на основі цих даних вивести усі сорти вказаної культури, які не завершили випробування на поточний рік.

3. На основі запитів сформувати звіти; заголовки цих звітів мають містити які вказує користувач, назви звітів мають чітко відображати їхній зміст.

4. У звіті за лабораторною роботою представити тексти запитів та сформовані звіти.

Приклад оформлення звітів

Форма «Експертні звіти» з кнопками усіх звітів зображено на рис.6.61.

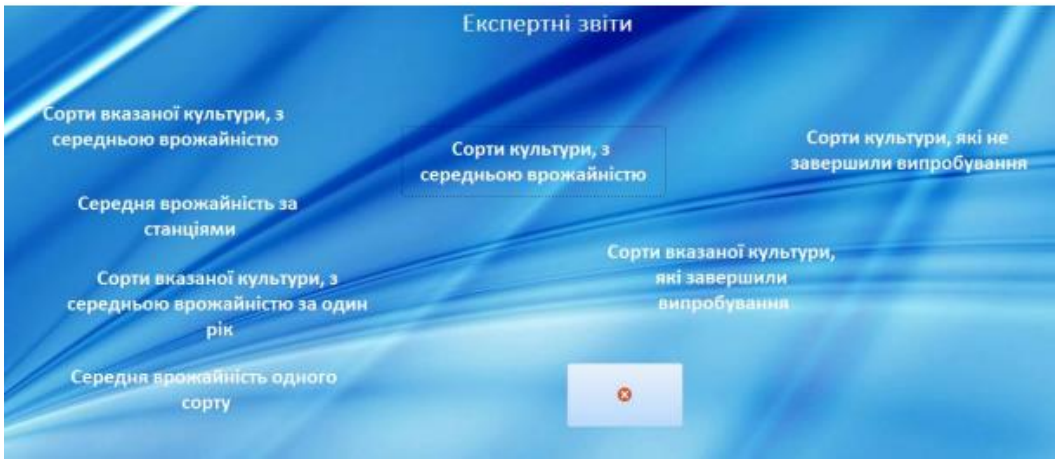


Рис.6.61 Форма «Експертні звіти»

Запит мовою SQL виконані за таким завданням:

1) користувач має вказати культуру, кліматичну зону, станцію, часовий період випробувань (роки); на основі цих даних вивести усі сорти вказаної культури, вказавши їхню середню врожайність за вказаний період, упорядковуючи за врожайністю представлено на рис.6.62, його виконання – на рис.6.63 та результат виконання – на рис.6.64;

```
SELECT DISTINCT Variety.Name_variety AS Сорти, Avg(Test.Crop_capacity) AS [Середня врожайність], Culture.Name_ua
FROM Variety, Heaven, Culture, Station, Test
WHERE Culture.Name_ua Like "*" & [Формы]![Form2]![Clture] & "*" AND Heaven.Name_heaven=[Формы]![Form2]![heav] AND
Station.Name_station Like "*" & [Формы]![Form2]![st] & "*" AND Test.Year Between [Формы]![Form2]![year1] And [Формы]![Form2]![year2]
AND Culture.[#C]=[Variety].[#C] AND Variety.[#C]=[Test].[#C] AND Variety.[#V]=[Test].[#V]
GROUP BY Variety.Name_variety, Culture.Name_ua
ORDER BY Avg(Test.Crop_capacity);
```

Рис. 6.62. Запит мовою SQL виконаний за завданням 1

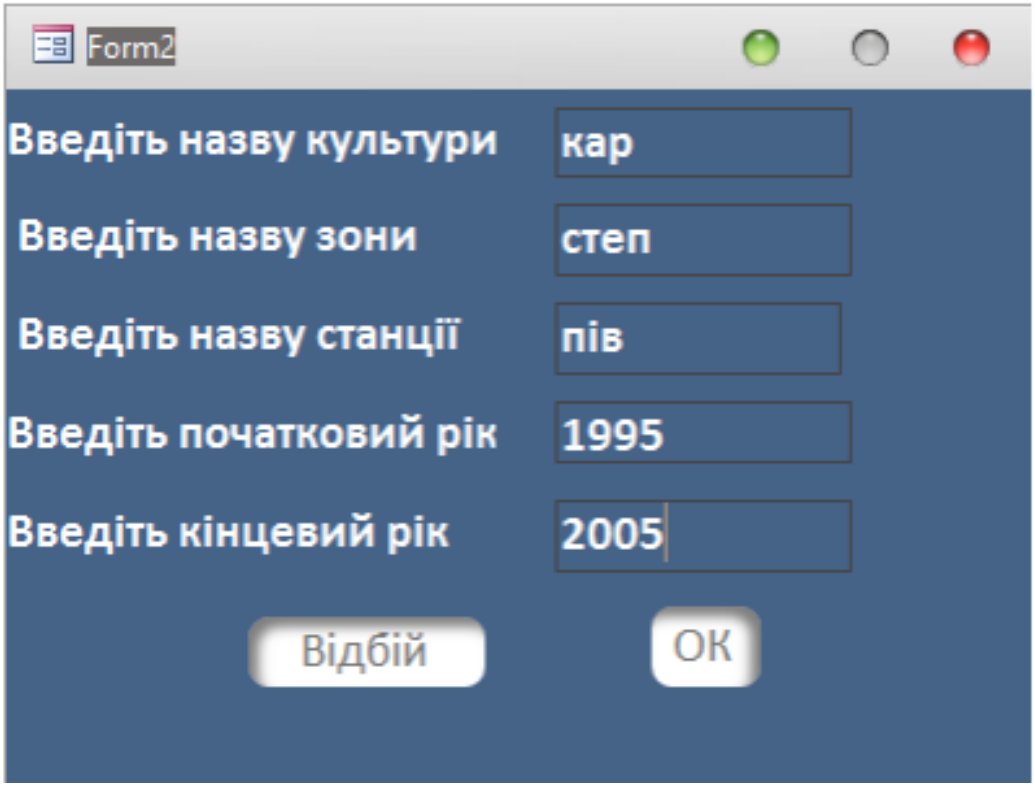


Рис. 6.63 Виконання запиту мовою SQL виконаний за завданням 1

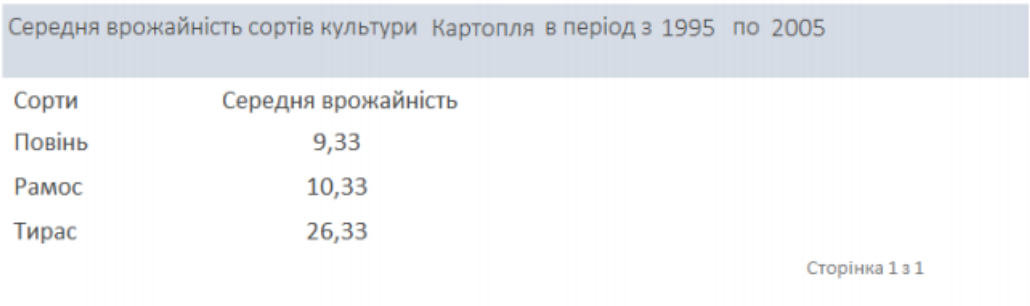


Рис. 6.64 Результат запиту мовою SQL виконаний за завданням 1

1а) за усіма станціям вказаної кліматичної зони представлено на рис.6.65, його виконання – на рис.6.66 та результат виконання – на рис.6.67;

```
SELECT DISTINCT Variety.Name_variety AS Сорти, Avg(Test.Crop_capacity) AS [Середня врожайність], Culture.Name_ua,
Station.Name_station, Heaven.Name_heaven
FROM Variety, Heaven, Culture, Station, Test
WHERE Culture.Name_ua Like "*" & [Формы]![Form5]![Clture] & "*" AND Heaven.Name_heaven=[Формы]![Form5]![heav] AND Test.Year
Between [Формы]![Form5]![year1] And [Формы]![Form5]![year2] AND Culture.[#C]=[Variety].[#C] AND Variety.[#C]=[Test].[#C] AND
Variety.[#V]=[Test].[#V] AND Test.[#S]=[Station].[#S] AND Heaven.[#H]=[Station].[#H]
GROUP BY Variety.Name_variety, Culture.Name_ua, Station.Name_station, Heaven.Name_heaven
ORDER BY Avg(Test.Crop_capacity);
```

Рис. 6.65 Запит мовою SQL виконаний за завданням 1а

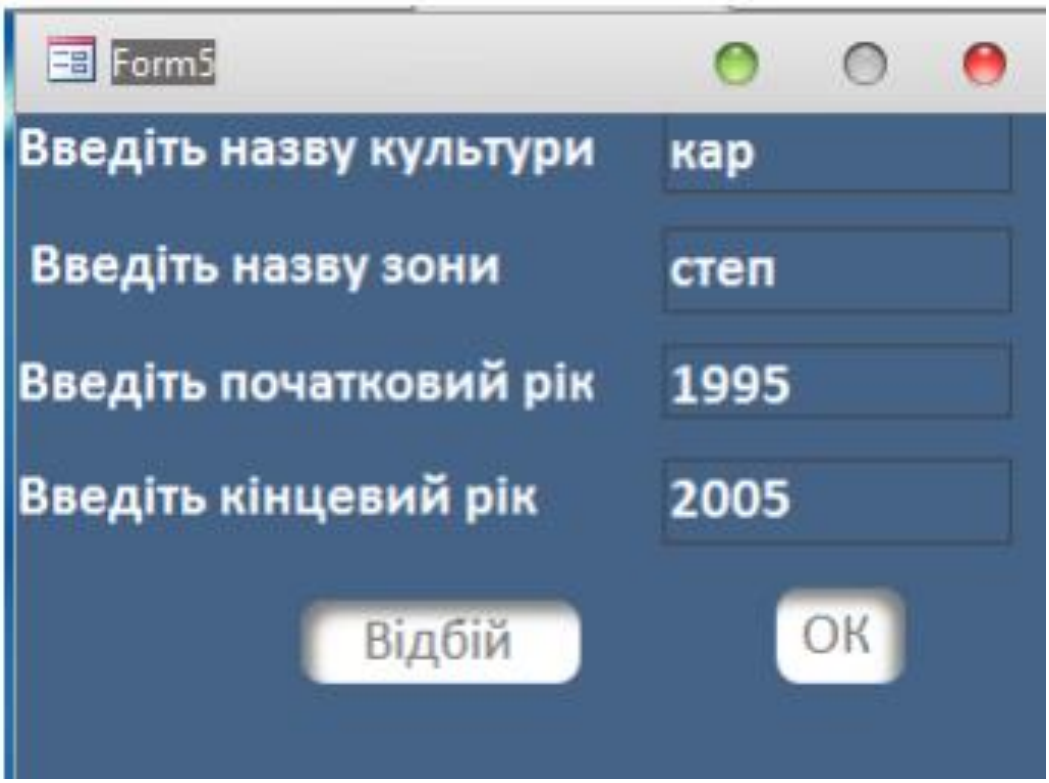


Рис. 6.66 Виконання запиту мовою SQL виконаний за завданням 1а

Середня врожайність сортів культури Картопля за станціями кліматичної зони Степ в період з 1995 по 2005

Сорти	Середня врожайність	Назва станції
Повінь	5,00	Південна дослідна станція
Рамос	7,00	Південна дослідна станція
Тирас	8,00	Південна дослідна станція

Сторінка 1 з 1

Рис. 6.67 Результат запиту мовою SQL виконаний за завданням 1а

1б) за один рік представлено на рис.6.68, його виконання – на рис.6.69 та результат виконання – на рис.6.70;

```
SELECT DISTINCT Variety.Name_variety AS Сорти, Avg(Test.Crop_capacity) AS [Середня врожайність], Culture.Name_ua,
Station.Name_station, Test.Year
FROM Variety, Heaven, Culture, Station, Test
WHERE Culture.Name_ua Like "*" & [Формы]![Form7]![Clture] & "*" AND Heaven.Name_heaven=[Формы]![Form7]![heav] AND Test.Year
Like "*" & [Формы]![Form7]![year] & "*" AND Culture.[#C]=[Variety].[#C] AND Variety.[#C]=[Test].[#C] AND Variety.[#V]=[Test].[#V] AND
Station.[#S]=[Test].[#S]
GROUP BY Variety.Name_variety, Culture.Name_ua, Station.Name_station, Test.Year
ORDER BY Avg(Test.Crop_capacity);
```

Рис. 6.68 Запит мовою SQL виконаний за завданням 1б

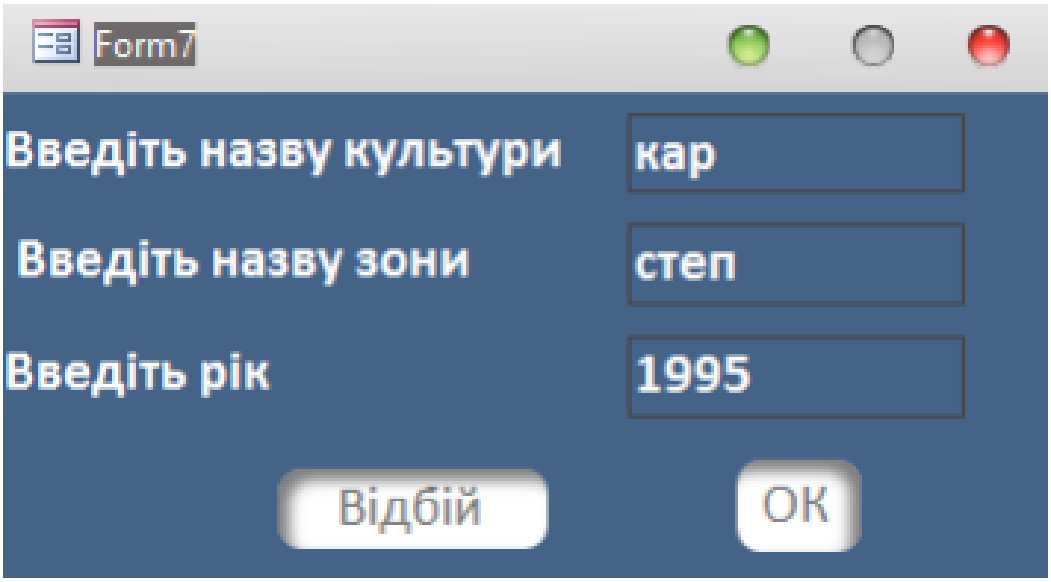


Рис. 6.69 Виконання запиту мовою SQL виконаний за завданням 1б

Сорти культури Картопля в 1995 році

Сорти	Середня врожайність	Назва станції
Повінь	5,00	Південна дослідна станція
Повінь	11,00	ДП ДГС ІАП НААН
Повінь	12,00	СКВИРСЬКА СТАНЦІЯ
Рамос	7,00	Південна дослідна станція
Рамос	8,00	ДП ДГС ІАП НААН
Рамос	16,00	Кіровоградська ДСГДС НААН
Тирас	15,00	Кіровоградська ДСГДС НААН
Тирас	56,00	ДОСЛІДНА СТАНЦІЯ "ЕЛІТА"

Сторінка 1 з 1

Рис. 6.70 Результат запиту мовою SQL виконаний за завданням 1б

1в) лише один сорт та комбінації цих варіантів представлено на рис.6.71, його виконання – на рис.6.72 та результат виконання – на рис.6.73;

```
SELECT DISTINCT Variety.Name_variety AS Сорти, Avg(Test.Crop_capacity) AS [Середня врожайність], Culture.Name_ua
FROM Variety, Heaven, Culture, Station, Test
WHERE Culture.Name_ua Like "*" & [Формы]![Form4]![Clture] & "*" AND Variety.Name_variety Like "*" & [Формы]![Form4]![variet] & "*"
AND Heaven.Name_heaven=[Формы]![Form4]![heav] AND Station.Name_station Like "*" & [Формы]![Form4]![st] & "*" AND Test.Year
Between [Формы]![Form4]![year1] And [Формы]![Form4]![year2] AND Culture.[#C]=[Variety].[#C] AND Variety.[#C]=[Test].[#C] AND
Variety.[#V]=[Test].[#V]
GROUP BY Variety.Name_variety, Culture.Name_ua
ORDER BY Avg(Test.Crop_capacity);
```

Рис. 6.71 Запит мовою SQL виконаний за завданням 1в

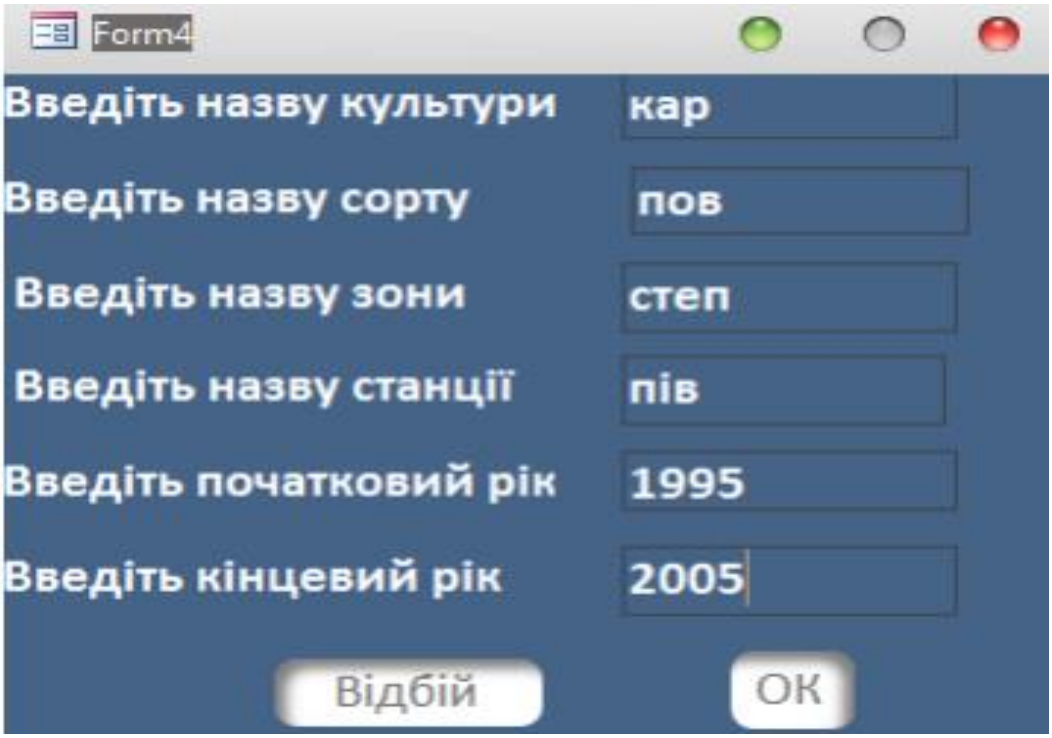


Рис. 6.72 Виконання запиту мовою SQL виконаний за завданням 1в

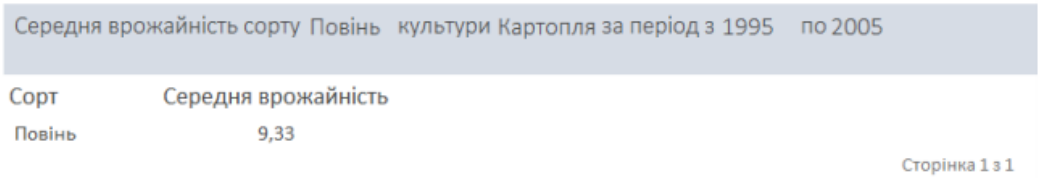


Рис. 6.73 Результат запиту мовою SQL виконаний за завданням 1в

2) користувач вказує назву культури, кліматичну зону, мінімальну та максимальну значення врожайності; на основі цих даних вивести лише ті сорти вказаної культури, які у поточному році завершили випробування та мають середню врожайність, що знаходиться у вказаному діапазоні врожайності представлено на рис.6.74, його виконання – на рис.6.75 та результат виконання – на рис.6.76;

```
SELECT DISTINCT Variety.Name_variety AS Сорти, Culture.Name_ua
FROM Variety, Heaven, Culture, Station, Test
WHERE Culture.Name_ua Like "*" & [Формы]![Form3]![Clture] & "*" AND Heaven.Name_heaven=[Формы]![Form3]![heav] AND
Test.Crop_capacity Between [Формы]![Form3]![min] And [Формы]![Form3]![max] AND Culture.[#C]=[Variety].[#C] AND
Variety.[#C]=[Test].[#C] AND Variety.[#V]=[Test].[#V] AND Year(Date())>[Year]
GROUP BY Variety.Name_variety, Culture.Name_ua
HAVING (((Avg(Test.Crop_capacity))<>False));
```

Рис. 6.74 Запит мовою SQL виконаний за завданням 2

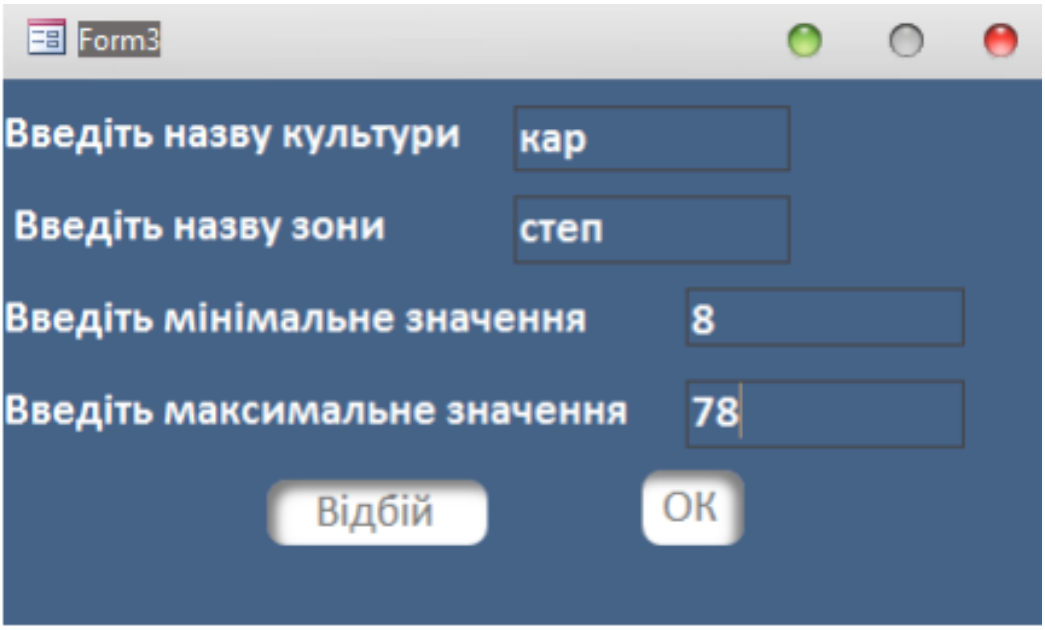


Рис. 6.75 Виконання запиту мовою SQL виконаний за завданням 2

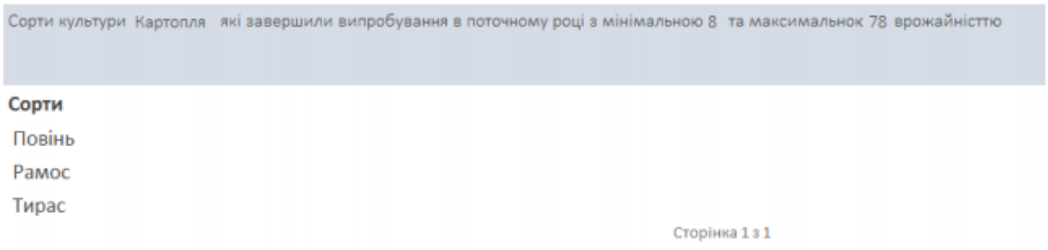


Рис. 6.76 Результат запиту мовою SQL виконаний за завданням 2

3) користувач вказує назву культури та часовий період випробування; на основі цих даних вивести усі сорти вказаної культури, вказавши їхню середню врожайність, об’єднавши їх в окремі групи відповідно кліматичних зонах представлено на рис.6.77, його виконання – на рис.6.78 та результат виконання – на рис.6.79;

```
SELECT DISTINCT Variety.Name_variety AS Сорти, Avg(Test.Crop_capacity) AS [Середня врожайність], Heaven.Name_heaven,
Culture.Name_ua
FROM Variety, Heaven, Culture, Station, Test
WHERE Culture.Name_ua Like "*" & [Формы]![Form1]![CIture] & "*" AND Test.Year Between [Формы]![Form1]![year1] And
[Формы]![Form1]![year2] AND Culture.[#C]=[Variety].[#C] AND Variety.[#C]=[Test].[#C] AND Variety.[#V]=[Test].[#V] AND
Heaven.[#H]=[Station].[#H] AND Station.[#S]=[Test].[#S]
GROUP BY Variety.Name_variety, Heaven.Name_heaven, Culture.Name_ua;
```

Рис. 6.77 Запит мовою SQL виконаний за завданням 3

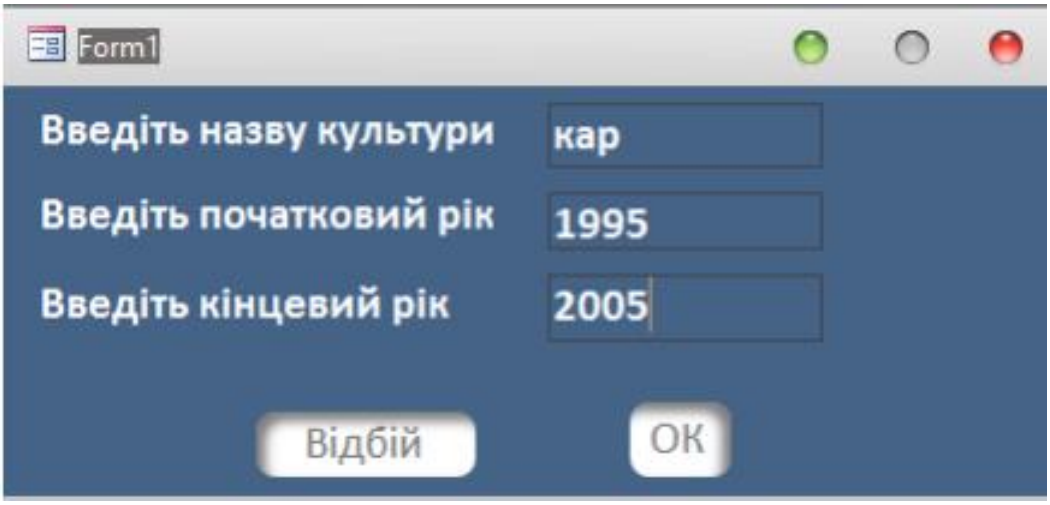


Рис. 6.78 Виконання запиту мовою SQL виконаний за завданням 3

Сорти культури Картопля з середньою врожайністю за період з 1995 по 2005

Сорти	Середня врожайність	Назва кліматичної зони
Повінь	5	Степ
Повінь	11	Полісся
Повінь	12	Лісостеп
Рамос	7	Степ
Рамос	8	Полісся
Рамос	16	Лісостеп
Тирас	8	Степ
Тирас	15	Лісостеп
Тирас	56	Полісся

Сторінка 1 з 1

Рис. 6.79 Результат запиту мовою SQL виконаний за завданням 3

4) користувач вказує назву культури; на основі цих даних вивести усі сорти вказаної культури, які не завершили випробування на поточний рік представлено на рис.6.80, його виконання – на рис.6.81 та результат виконання – на рис.6.82;

```
SELECT DISTINCT Variety.Name_variety AS Сорти, Culture.Name_ua
FROM Variety, Heaven, Culture, Station, Test
WHERE Culture.Name_ua Like "*" & [Формы]![Form6]![Clture] & "*" AND Culture.[#C]=[Variety].[#C] AND Variety.[#C]=[Test].[#C] AND
Variety.[#V]=[Test].[#V] AND Test.Year=Year(Date())
```

Рис. 6.80 Запит мовою SQL виконаний за завданням 4

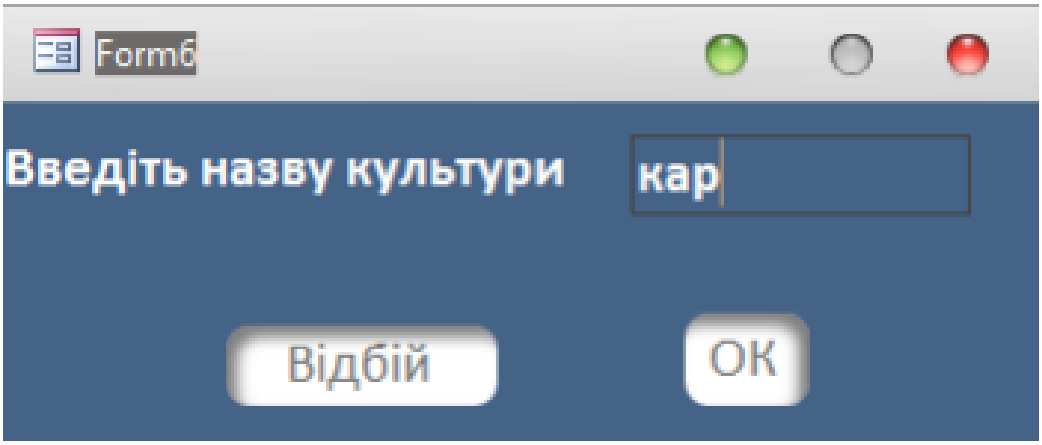


Рис. 6.81 Виконання запиту мовою SQL виконаний за завданням 4

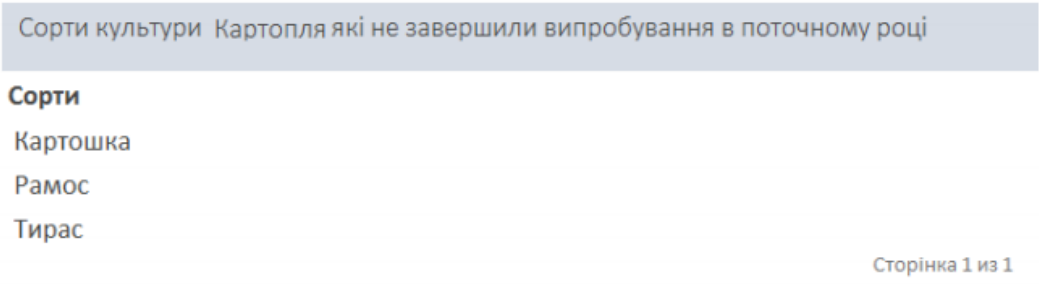


Рис. 6.82 Результат запиту мовою SQL виконаний за завданням 3

6.8 Побудова інтерфейсу користувача засобами C++Builder

Теоретична частина

До цієї лабораторної роботи ми розробляли інтерфейсну частину проекту у середовищі MS Access. Поставимо за мету реалізувати повноцінний інтерфейс користувача засобами C++Builder.

Кроки, які необхідно зробити для виведення змісту таблиці бази даних, представлені нижче.

Крок 1. *Створення нового проекту*. Після завантаження C++Builder створіть новий проект як на рис. 6.83.

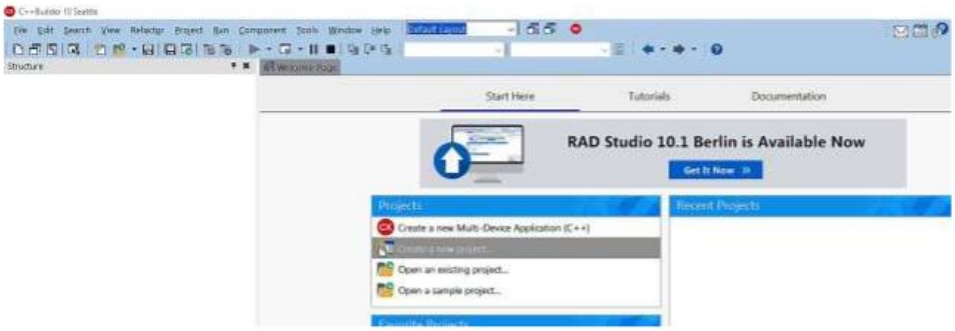


Рис.6.83 Створення нового проекту

Крок 2. *Вибір типу проекту*. Вибираємо ти проекту «VCL Form Application» (рис. 6.84).

Рис.6.84 Вибір типу проекту

Крок 3. *Оформлення головного вікна*. Для визначення заголовку вікна у вікні «Object Inspector» вибираємо властивість «Caption» (рис.6.85).

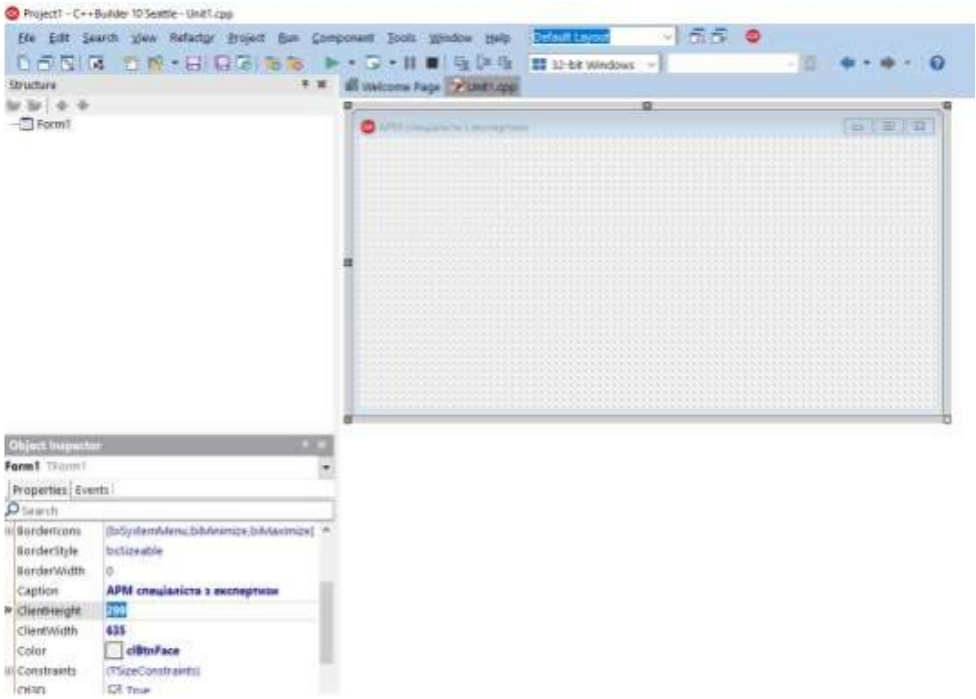


Рис.6.85 Оформлення головного вікна

Крок 4. *Додаємо до проекту ще один файл* – Data Module (рис.6.86).

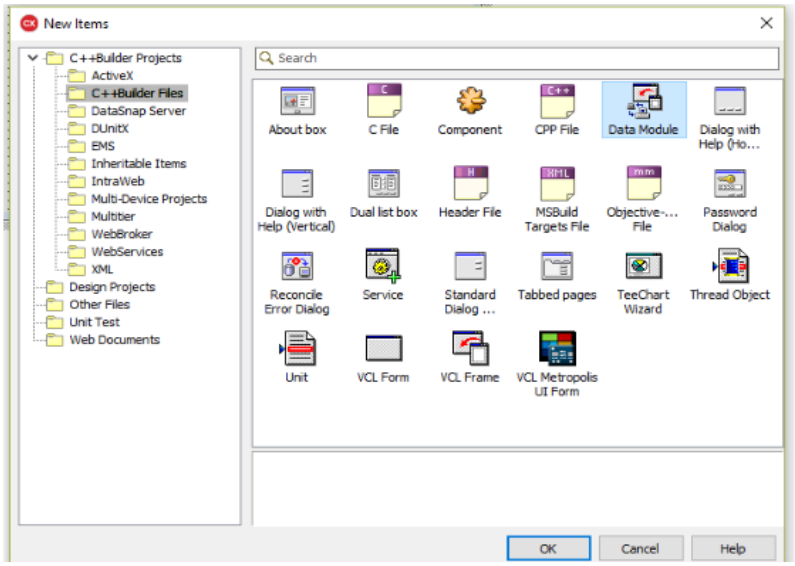


Рис.6.86 Додавання до проекту файлу

Крок 5. *Додавання компонентів у вікно Data Module*. У вікні «Tool Palette» обираємо вкладку «DBGo», з осередка якої переносимо до форми необхідні компоненти (рис.6.87).

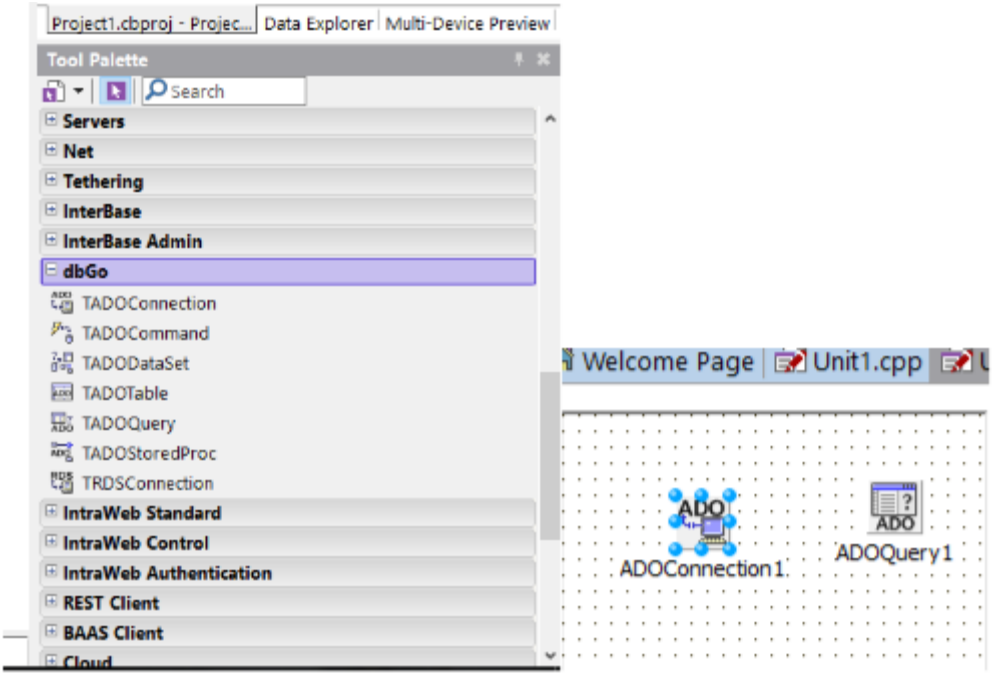


Рис.6.87 Додавання компонентів у вікно Data Module

Крок 6. *Визначення параметрів зв'язку з базою даних*. Вибираємо компонент TADOConnection на формі, у вікні «Object Inspector» - властивість «Connection String», за допомогою майстра створюємо рядок з'єднання, який має бути як на рис.6.88.

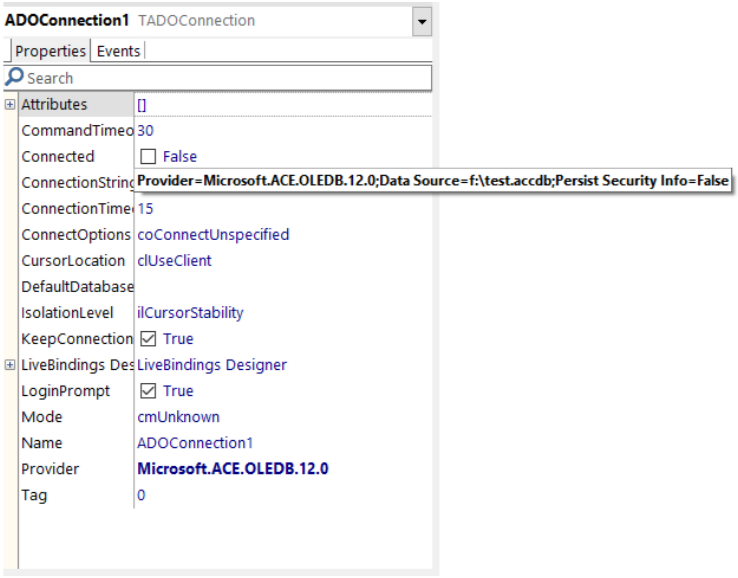


Рис.6.88 Визначення параметрів зв'язку з базою даних

Крок 7. *Формування запиту для виведення рядків з таблиці*. Вибираємо компонент TADOQuery на формі, у вікні «Object Inspector» - властивість «Connection», для якої встановлюємо з'єднання з базою

даних через попередній компонент. Далі вибираємо властивість «SQL», в осередку якої записуємо запит на вибірку (рис.6.89).

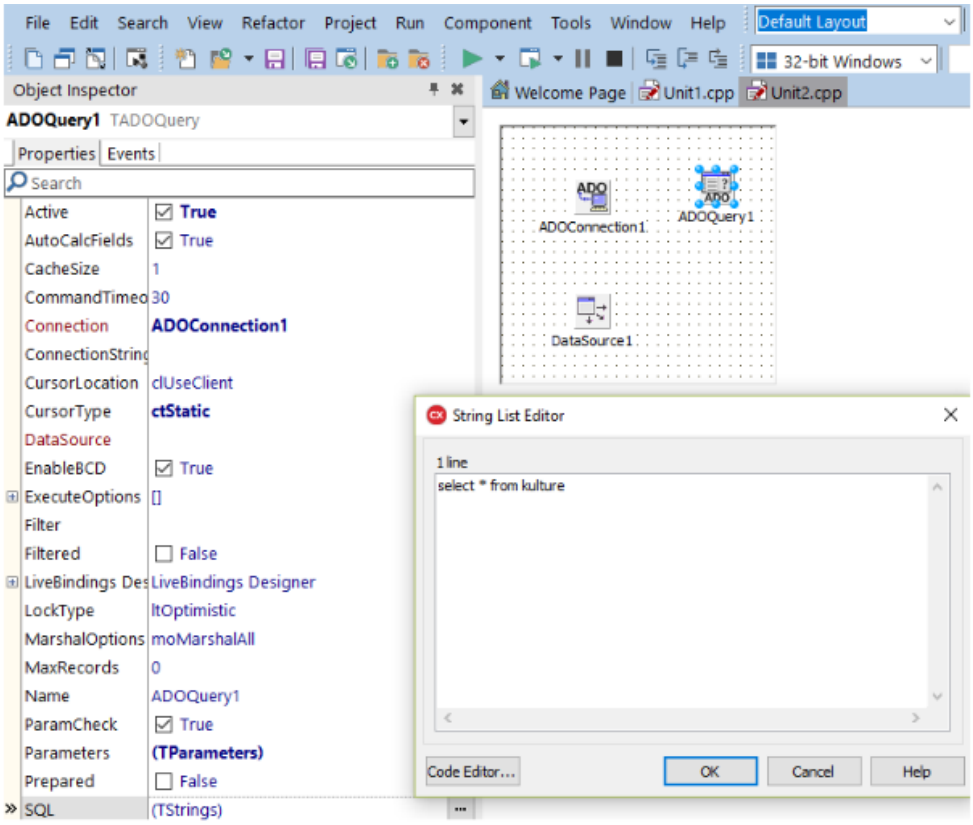


Рис.6.89 Формування запиту для виведення рядків з таблиці

Крок 8. *Додавання компонента TDataSource*. Останній компонент, який ми додаємо у вікно Data Module, розміщений у вкладці Data Access. Для нього також встановлюємо за допомогою властивості DataSet з компонентом TADOQuery (рис.90).

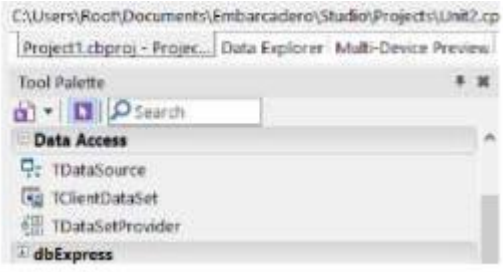


Рис.6.90 Додавання компонента TDataSource

Крок 9. *Виведення вмісту таблиці у вікно головної форми*. Для реалізації цієї задачі спочатку необхідно вставити у файл .cpp головної форми заголовний файл Data Module. Далі додаємо на головну форму

компонент dbGrid (вкладка Data Controls) і встановлюємо зв’язок з компонентом *TDataSource* (рис.6.91).

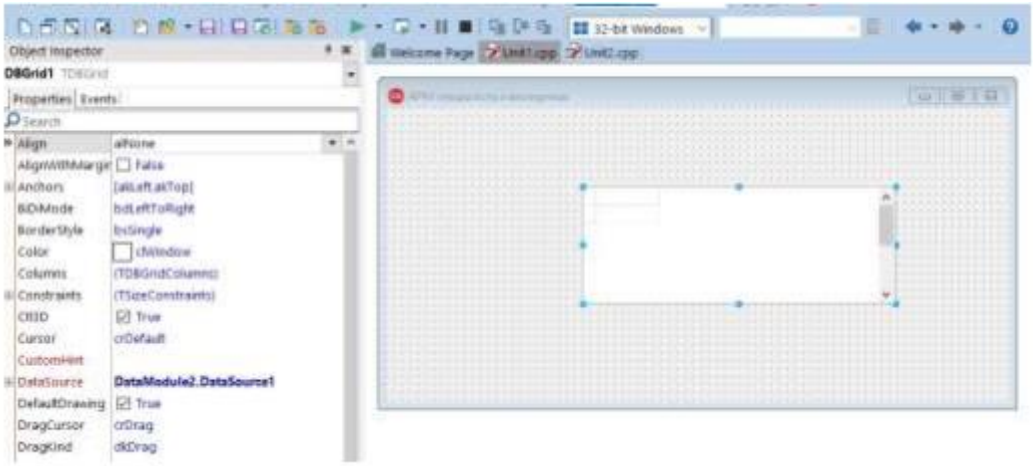


Рис.6.91 Виведення вмісту таблиці у вікно головної форми

Постановка завдання Основна мета лабораторної роботи – розробка інтерфейсу користувача для перегляду вмісту кожної з таблиць. Для цього Вам потрібно розробити відповідні запити та записати їх в осередку компонента TADOQuery. Крім того, Вам знадобиться розповсюдження на форми додаткових елементів.

Завдання на виконання лабораторної роботи

1. Розробити головну форму «АРМ спеціаліста з експертизи» та представити в її осередку у вигляді кнопок (меню, вкладок) усі таблиці.

2. Розробити форму Data Module для реалізації отримання доступу до таблиць.

3. У звіті за лабораторною роботою представити скрін-шоти.

Теми для самостійного вивчення

2. Історія розвитку СУБД: огляд різних моделей баз даних і відповідних СУБД у розрізі часу.

3. База даних, база знань, банк даних. Порівняльний аналіз.

4. Об'єктний підхід до структурізації даних у базі даних.

5. Правила Кодда для реляційної бази даних.

6. СУБД MS Access: імпорт та експорт даних.

7. СУБД MS Access: вбудовані функції.

8. СУБД MS Access: макрокоманди, макроси.

9. СУБД MS Access: VBA.

9. Історія виникнення мови SQL

10. Загальні можливості SQL; категорії команд

11. Використання складених запитів та приклади для учбового проекту

12. Приклади використання команди додавання запису у таблиці учбового проекту

13. Приклади використання команди видалення запису у таблиці учбового проекту

14. Приклади використання команди оновлення полів у таблиці учбового проекту

Бібліографічний список

1. Т.Карпов. Базы данных: модели, разработка, реализация. Учебник / Т. Карпов. – С.Петербург:”Питер”, 2001. – 304 с.

2. А.Д.Хомоненко. Базы данных. Учебник для ВУЗов. 2-е издание / Хомоненко А.Д. – С.Петербург:”Питер”, 2001. – 672 с.

3. Гектор Гарсиа-Молина. Системы баз данных. Полный курс / Гектор Гарсиа-Молина, Джеффри Д. Ульман, Дженнифер Видом. – Москва, Санки-Петербург, Киев: «Издательский дом ВИЛЬЯМС» , 2003. – 1088 с.

4. К.Дж.Дейт. Введение в системы базы данных / К.Дж.Дейт. – Москва, Санки-Петербург, Киев: «Издательский дом ВИЛЬЯМС» , 2005. – 1325 с.

5. Пасічник В. В. Організація баз даних і знань / Пасічник В. В., Резніченко В. А. – BHV, Киев, 2006. – 384 с.