Міністерство освіти і науки України
Чернівецький національний університет
імені Юрія Федьковича

С.А. Іліка, Л.А. Піддубна

БАЗИ ДАНИХ ТА ІНФОРМАЦІЙНІ СИСТЕМИ

Навчальний посібник

Чернівці
Чернівецький національний університет
2025

ББК 32.973.2–018я73

С–409

УДК 681.3.066(075.8)

Друкується за ухвалою Вченої Ради факультету математики та інформатики Чернівецького національного університету імені Юрія Федьковича, протокол №6 від 25.01.2023 р.

Рецензенти: *В.Г. Фратавчан*, канд. фіз.-мат. наук, доцент, доцент кафедри МПУіК Чернівецького національного університету імені Юрія Федьковича

Г.В. Мельник, канд. економ. наук, доцент, доцент кафедри прикладної математики та інформаційних технологій Чернівецького національного університету імені Юрія Федьковича, Senior Softserve

С-409 **БАЗИ ДАНИХ ТА ІНФОРМАЦІЙНІ СИСТЕМИ**. Навчальний посібник/ Укл.: Іліка С.А., Піддубна Л.А. Чернівці: Чернівецький нац. ун-т, 2025. 192 с.

Посібник містить теоретичний матеріал, рекомендації до виконання лабораторних робіт. Для студентів спеціальностей галузі інформаційних технологій.

УДК 512+514.12](075.8)

Вступ

База даних (БД) – це значна кількість однорідних даних з конкретної предметної галузі, які зберігаються на комп`ютерних носіях.

БД створюють, якщо є потреба регулярно опрацьовувати великі обсяги однорідної інформації: списки абітурієнтів, студентів з їхніми оцінками, співробітників підприємства чи фірми з анкетними даними, розклади руху різних видів транспорту, пропозиції товарів на ринку, облік матеріалів на складах тощо.

Робота з БД має такі етапи:

1) створення структури БД;
2) введення даних;
3) редагування структури і даних;
4) відшукання інформації в БД;
5) оформлення звітів.

Для виконання цих робіт є спеціальні програми, такі як dBase-системи, FoxPro, Access та ін. Вони називаються системами керування базами даних (СКБД).

В Access *база даних* – це файл, який містить дані у вигляді однієї чи кількох таблиць. Окрім таблиць, у файлі БД можуть бути такі об`єкти: форми, запити, звіти, макроси, модулі.

Таблиці БД складаються з рядків (записів) і стовпців.

Запис містить інформацію про один елемент БД: одну людину, книжку, продукцію, рейс тощо. Він складається з полів, які формують структуру запису. *Структура запису* фактично визначає структуру таблиці і всієї БД, якщо в ній є лише одна таблиця.

Поле – це мінімальна (але найважливіша) порція інформації в записі, над якою визначені операції введення, виведення, перетворення тощо. Воно має ім`я, значення, характеризується типом і низкою додаткових властивостей.

Назви полям дає користувач, назви типів стандартні, а значення полів випливають зі змісту конкретної задачі.

Отже, *структура таблиці* – це структура запису, тобто сукупність назв полів, їхніх типів та властивостей, визначених користувачем під час аналізу конкретної задачі. Структура визначає послідовність розташування даних у записі на фізичному носії і вигляд даних на екрані.

Список лабораторних робіт

1. Розробка структури бази даних. Застосування операцій реляційної алгебри
2. Реалізація схеми бази даних у MS Access.
3. Створення запитів у середовищі MS Access
4. Створення звітів у середовищі MS Access
5. Інтерфейс бази даних у середовищі MS Access.
6. Розробка бази даних у середовищі mySQL
7. Розробка запитів до бази даних у mySQL.

Частина 1. Інформаційні системи. Бази даних. Основні поняття та означення.

§ 1. Інформаційні системи. Їх класифікації та типи

1.1. Інформація й дані

Перш ніж перейти до обговорення поняття інформаційної системи (ІС), спробуємо з'ясувати, що ж розуміється під словом "інформація". Відповісти на це питання і просто, і складно: слово "інформація" пов'язане із широким колом понять.

Змістовна сторона поняття "інформація" дуже багатогранна й не має чітких семантичних меж. Однак завжди можна сказати, що можна з нею робити. Саме відповідь на це питання найчастіше й цікавить як системних аналітиків і розроблювачів ІС, так і користувачів інформації (її основних споживачів).

З погляду як користувачів, так і розроблювачів ІС в інформації є одна важлива властивість - вона є одиницею даних, яка підлягає обробці. Звичайно інформація надходить споживачеві саме у вигляді даних: таблиць, графіків, малюнків, фільмів, усних повідомлень, які фіксують у собі інформацію певної структури й типу. Таким чином, дані виступають як засіб подання інформації у певній, фіксованій формі, придатній для обробки, зберігання й передачі. Хоча дуже часто терміни "інформація" й "дані" виступають як синоніми, варто пам'ятати про цю їхню істотну відмінність. Саме в даних інформація знаходить інтерпретацію у конкретній ІС.

При згадуванні про "форму" подання інформації варто сказати ще про одну, "людську" властивість інформації - її сприйняття різними категоріями людей. Дані можуть бути згруповані спільно у документ. Документ може мати або не мати певної внутрішньої структури. Дані можуть бути відображені на екрані дисплея комп'ютера. Документи можуть мати аудіо- або відеоформу. Розробляючи ІС, ніколи не слід забувати, для кого вони (системи) створюються і хто буде їх використовувати. Форма подання інформації в ІС

визначає також і категорії користувачів. ІС створюються для конкретних груп користувачів, тобто вони, як правило, проблемно-орієнтовані.

Інформація є даними, яким надається деякий зміст (інтерпретація) у конкретній ситуації у рамках деякої системи понять. Інформація подається за допомогою кодування даних і здобувається шляхом їхнього декодування та інтерпретації.

У цьому визначенні фіксується три основних перетворення інформації й даних у процесі їхньої обробки в ІС: інформація – дані, дані – дані, дані – інформація.

На рис. 1.1 наведені дві сторони визначення поняття інформації: функціональна й представницька. Перша загалом визначає коло дій над інформацією, а друга – результат виконання цих дій.

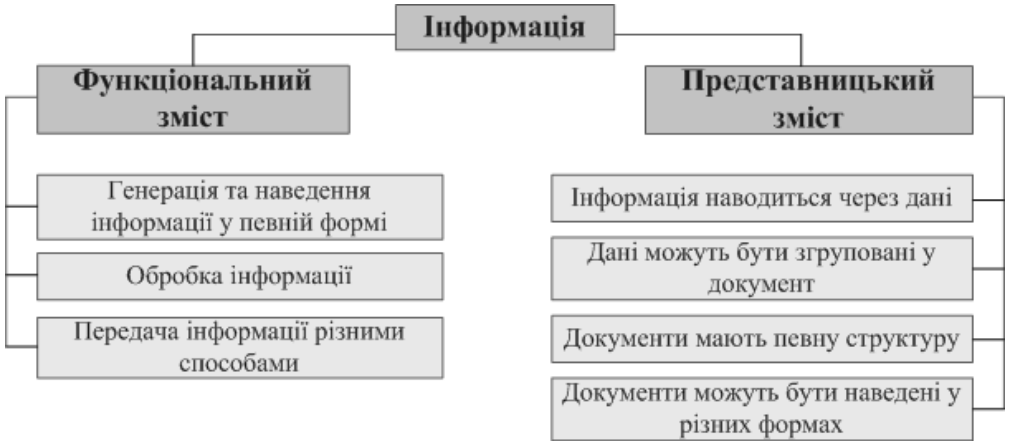


Рисунок 1. Зміст поняття "інформація"

1.2. Інформаційні системи

Основною метою створення ІС є задоволення інформаційних потреб користувачів шляхом надання необхідної їм інформації на основі збережених даних. Потреба в інформації як такій не вичерпує поняття інформаційних потреб. Звичайно в поняття інформаційних потреб включають певні вимоги до якості інформаційного обслуговування й поводження системи в цілому (продуктивність, актуальність і надійність даних, орієнтація на користувача та ін.).

Під інформаційною системою розуміється організаційна сукупність технічних засобів, технологічних процесів і кадрів, що реалізують функції збору, обробки, зберігання, пошуку, видачі й передачі інформації.

Необхідність підвищення продуктивності праці у сфері інформаційної діяльності приводить до того, що в якості зовнішніх засобів зберігання й швидкого доступу до інформації найчастіше використовуються засоби обчислювальної техніки (цифровий і аналоговий) на основі комп'ютерів. Сучасні ІС - складні комплекси апаратних і програмних засобів, технології й персоналу, які ще називають автоматизованими інформаційними системами. Структурно ІС містять у собі апаратне (hardware), програмне (software), комунікаційне (netware), проміжного шару (middleware), лінгвістичне й організаційно-технологічне забезпечення.

Апаратне забезпечення ІС містить у собі широкий набір засобів обчислювальної техніки, передачі даних, а також цілий ряд спеціальних технічних пристроїв (пристрою графічного відображення інформації, аудио- і відеопристрою, засобу мовного введення й т.д.). Апаратне забезпечення є основою будь-якої ІС.

Комунікаційне (мережне) забезпечення містить у собі комплекс апаратних мережних комунікацій і програмних засобів підтримки комунікацій в ІС. Воно має істотне значення при створенні розподілених ІС й ІС на основі Інтернету.

Програмне забезпечення ІС забезпечує реалізацію функцій введення даних, їх розміщення на носіях, модифікації даних, доступ до даних, підтримку функціонування устаткування. Програмне забезпечення можна розділити на системне (яке завершує процес вибору апаратно-програмного рішення або платформи) і користувацьке (яке застосовується для вирішення завдань задоволення потреб користувача у комп'ютерному середовищі).

Лінгвістичне забезпечення ІС призначене для вирішення завдань формалізації змісту повнотекстової і спеціальної інформації для створення пошукового образу даних (профілю). У класичному змісті звичайно воно

включає процедури індексування текстів, їхню класифікацію і тематичну рубрикацію. Найчастіше ІС, що містять складно-структуровану інформацію, містять у собі тезауруси термінів і понять. Сюди можна віднести й створення процесорів спеціалізованих формальних мов кінцевих користувачів, наприклад, мов для маніпулювання бухгалтерською інформацією і т.д. Найчастіше працям з розроблення лінгвістичного забезпечення не надається належного значення. Подібні недогляди найчастіше призводять до несприйняття користувачами самої інформації. Це стосується в першу чергу вузько спеціалізованих ІС.

У міру зростання складності і масштабів ІС важливу роль починає відігравати організаційно-технологічне забезпечення, що з'єднує різнорідні компоненти (апаратури, програми й персонал) у єдину систему й забезпечує процедури її керування й функціонування. Недооцінка цієї складової ІС найчастіше призводить до зриву строків впровадження системи й виведення її на виробничі потужності.

На рис. 2 наведені функції ІС через її основні структурні компоненти.

Рисунок 2. Визначення інформаційної системи

1.3. Основні підходи до обробки інформації в автоматизованих інформаційних системах

Одним з головних питань розроблення програмного забезпечення ІС є питання про співвіднесення програм і даних, тому що вирішення цього питання, в остаточному підсумку, визначає вибір алгоритмів обробки інформації, апаратних засобів і технологічної платформи. Фундаментальним

принципом у вирішенні питання про співвіднесення програм і даних є концепція незалежності прикладних програм від даних, і неважливо, яка обробка даних передбачається: централізована або розподілена. Суть цієї концепції полягає не стільки у відділенні програм від даних, скільки у розгляді їх як самостійних взаємодіючих об'єктів.

Однією з останніх модифікацій цього принципу є концепція незалежності прикладних програм від даних разом із процедурами їхньої обробки (об’єктно-орієнтований підхід у програмуванні), що дозволяє вирішити ряд питань обробки даних, пов'язаних з інтерпретацією семантичного змісту даних.

Формування концепції БД і створення на її основі методу баз даних для вирішення завдань обробки інформації відбулося у 1962 році. До середини 60-х років минулого століття основною концепцією побудови програмного забезпечення були концепція файлової системи і так званий позадачний метод. Наприкінці 80-х років минулого століття була запропонована концепція об’єктно-орієнтованих баз даних й об’єктно-орієнтований підхід розроблення програм на основі обробки подій. На рис.1.3 наведені основні ознаки для кожної з зазначених вище концепцій. На рис. 1.4 проведене зіставлення основних методів обробки даних.

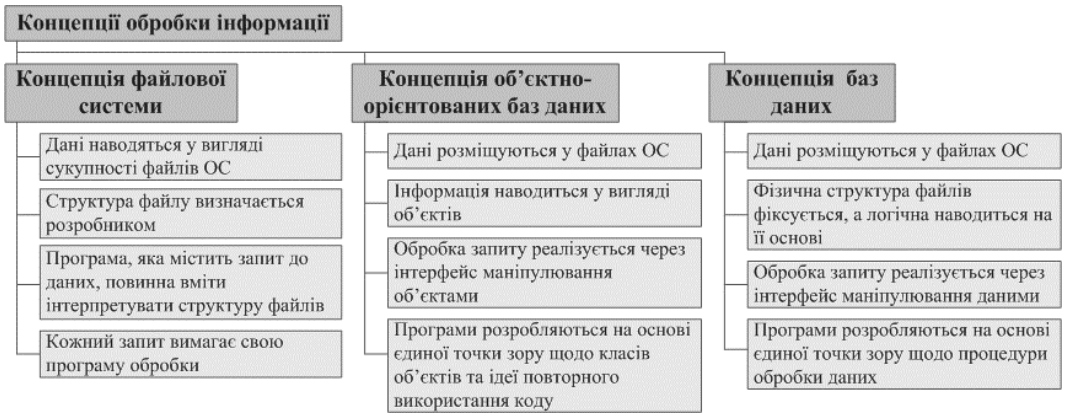


Рисунок 3. Основні концепції обробки інформації

Рисунок 4. Основні проблеми методів обробки інформації

Основний зміст позадачного методу зводиться до декомпозиції програми зі своїми окремими блоками даних та алгоритмами; методу баз даних – до наявних окремих описів логічної структури даних та єдиної точки зору щодо процедури обробки даних; об'єктно- орієнтованого методу – полягає в тому, що програми розглядаються як сукупність об'єктів, між якими відбувається обмін інформацією.

Об'єкту притаманні такі властивості:

- інкапсуляція – об'єкти наділяються структурою й мають певне поводження (набором операцій). Операції над об'єктами становлять його методи. Структура об'єкта захована від користувача, що маніпулює об'єктом через його операції. Об'єкт розглядається як абстракція реального світу Для того щоб об'єкт виконав деяку дію, йому потрібно надіслати повідомлення. Об'єкт взаємодіє з іншими об'єктами через події;
- спадкування – являє собою механізм, що дозволяє робити одні об'єкти з інших, при цьому властивості батьківського об'єкта зберігаються у нащадка;
- поліморфізм – різні об'єкти можуть одержувати однакові повідомлення, але реагувати на них по-різному відповідно до реалізації своїх однойменних методів.

Отже, при розробленні автоматизованих ІС слід пам’ятати:

- ІС створюються для конкретних груп користувачів, тобто вони є проблемно-орієнтовані;

- основною метою створення ІС є задоволення інформаційних потреб користувачів шляхом надання необхідної їм інформації на основі збережених даних;
- сучасні ІС - складні комплекси апаратних і програмних засобів, технології й персоналу, які ще називають автоматизованими інформаційними системами;
- дотримання концепції незалежності прикладних програм від даних разом із процедурами їхньої обробки (об’єктно-орієнтований підхід у програмуванні) дозволяє вирішити ряд питань обробки даних, пов'язаних з інтерпретацією семантичного змісту даних;
- ІС працює з упорядкованими взаємозалежними даними, які зберігаються з мінімальною надлишковістю та становлять базу даних. Системою керування базами даних називається сукупність програмних засобів, необхідних для використання БД і подання розробникам і користувачам безлічі різних подань даних.

§2. Бази даних. Системи керування базами даних

2.1. Бази даних і системи керування базами даних

Базу даних у загальному випадку можна визначити як уніфіковану сукупність збережених і відтворених даних, що використовуються у рамках організації (Engles R.A., 1972 р.). Однак поняття БД не ґрунтується в цей час на єдиній концепції, скоріше це ціле сімейство пов'язаних між собою понять з ПО, програмного й апаратного забезпечення, аналізу й моделювання даних і додатків. Існує кілька визначень БД.

База даних (за Дж. Мартіном) є сукупністю взаємозалежних даних, які спільно використовуються декількома додатками й зберігаються з мінімальною регульованою надлишковістю. Дані запам'ятовуються таким чином, щоб вони у міру можливості не залежали від програм. Для обробки даних застосовується загальний керуючий метод доступу. Якщо БД не перетинаються за структурою, то говорять про систему баз даних.

База даних (відповідно до матеріалів комітету КОДАСІЛ) складається зі всіх екземплярів записів, екземплярів наборів записів і областей, які контролюються конкретною схемою. Під схемою можна розуміти карту всієї логічної структури БД.

Для розроблювача ІС істотним моментом при використанні концепції баз даних (БД) є та обставина, що дані стають певним чином організовані, здобувають якусь упорядкованість і внутрішню структуру, а також те, що є деякий набір уніфікованих операцій обробки даних і декларативних засобів подання даних. До таких операцій варто віднести операції "Вставити" (Insert), "Додати" (Add), "Видалити" (Delete) і ряд інших. До декларативних засобів подання даних варто віднести мови визначення даних. Тобто використання даної концепції при створенні ІС припускає наявність мови визначення даних і мови маніпулювання даними, а також правил побудови інтерфейсів програм (додатків)із БД і користувачем.

Такий розподіл засобів маніпулювання даними і їхнього подання є деякою мірою умовним. Мова визначення даних служить для опису логічної

структури (схеми) БД, а в деяких випадках і способів зберігання й доступу до даних. Мова маніпулювання даними надає алгоритмічні засоби побудови додатків для обробки елементів даних, які зберігаються в БД.

2.2.Системи керування базами даних

У разі застосування концепції БД для створення ІС природно виникає питання: хто або що повинне все це підтримувати? Таким чином, постає питання про Систему керування базою даних (СКБД). СКБД є складними програмними системами, що працюють на різних операційних платформах. Саме СКБД повинна надати засоби визначення й маніпулювання даними, зробивши дані незалежними від прикладних програм, що їх використовують.

До основних функцій СКБД слід віднести:

- забезпечення мовних засобів опису та маніпуляції даними;
- забезпечення підтримки логічної моделі даних;
- забезпечення взаємодії логічної та фізичної структур даних;
- забезпечення захисту та цілісності даних;
- забезпечення підтримки БД в актуальному стані.

Системою керування базами даних (Data-base Management System) називається сукупність програмних засобів, необхідних для використання БД і подання розробникам і користувачам безлічі різних подань даних.

2.3. Порівняльна характеристика популярних СУБД

Microsoft Access

Особливості: файлова СУБД (Jet/ACE), інтегрована з MS Office, GUI для дизайну таблиць/запитів/форм.

Переваги: дуже швидкий старт для прототипів і малих додатків; зручний інтерфейс для нефахівців; локальне зберігання (файл .accdb).

Недоліки: не підходить для навантажень/багатокористувацьких виробничих систем; обмеження по розміру файлу; слабка масштабованість і безпека; погана підтримка реплікації/кластеризації.

Доречно використовується для : невеликих офісних рішень, у швидких прототипах, у навчанні.

MySQL

Особливості: популярна open-source СУБД, кілька движків зберігання (InnoDB — ACID). Часто використовується в веб-стеках (LAMP). **Переваги:** проста в налаштуванні, велика екосистема, багато хостингів підтримують; добра продуктивність для читання; сильна спільнота. **Недоліки:** історично мала сумісність зі стандартом SQL у деяких місцях; деякі функції (наприклад, повноцінна реалізація MVCC на старих движках) відрізняються від PostgreSQL; складніші фічі (широкі аналітичні запити) поступаються.

Доречно використовується для: веб-додатки, CMS, e-commerce.

MariaDB

Особливості: форк MySQL, сумісний, але додає нові движки й фічі; відкрита модель розвитку.

Переваги: сумісність із MySQL + додаткові оптимізації і движки; активний розвиток.

Недоліки: розгалуження екосистеми (іноді плутанина сумісності з MySQL-фічами); кілька різних випусків/версій. **Кейси:** як заміна MySQL там, де потрібні додаткові рухомі можливості.

Microsoft SQL Server (MS SQL)

Особливості: потужна корпоративна СУБД від Microsoft; тісна інтеграція з .NET, Windows, BI інструментами (SSRS/SSIS/SSAS). Підтримує як Windows, так і Linux.

Переваги: багаті інструменти адміністрування, гарна продуктивність у OLTP, масштабованість, підтримка транзакцій/ACID, безпека, корпоративні інструменти для аналітики.

Недоліки: ліцензування (дороге для великих середовищ), деякі фічі найкраще працюють у Windows-екосистемі.

Доречно використовується для: корпоративних додатків, BI, фінансових систем.

PostgreSQL

Особливості: “функціонально-багата” open-source СУБД з високою відповідністю стандартам SQL, багатовимірними типами, розширеннями (PostGIS, PL/pgSQL, тощо), MVCC.

Переваги: дуже висока сумісність зі стандартом, потужні можливості розширення (додаткові типи даних, індекси, функції), сильна підтримка транзакцій і одночасності, підходить як для OLTP, так і для аналітики, стабільність і надійність.

Недоліки: інколи складніша тюнінгова конфігурація; в деяких специфічних високонавантажених читальних сценаріях MySQL може бути швидшим “з коробки”.

Доречно використовується для: у корпоративних проектах, геопросторових системах, системах із складною бізнес-логікою.

Oracle Database

Особливості: одна з найпотужніших корпоративних СУБД; багатий функціонал (паралельні запити, advanced security, Data Guard, RAC). Платна, але з потужними можливостями для масштабування й відмовостійкості.

Переваги: дуже висока продуктивність для великих навантажень, розширені можливості масштабування й відновлення, багата функціональність для корпоративного рівня, сильні інструменти адміністрування.

Недоліки: висока вартість ліцензій і підтримки; складність адміністрування; закрита екосистема.

Доречно використовується для: великих банківських систем, телеком, ERP-систем, критичні корпоративні бази.

Firebird

Особливості: легка open-source СУБД, походить від InterBase; підтримує як вбудований (embedded), так і серверний режим. Підтримує транзакції та SQL.

Переваги: проста у використанні, невеликі вимоги до ресурсів, стабільна, добре підходить для вбудованих/локальних рішень. **Недоліки:** менша екосистема/ком'юніті та менше сучасних функцій у порівнянні з PostgreSQL чи MySQL; менш розвинена підтримка інструментів. **Доречно використовується для:** у вбудованих додатках, десктопних системах, невеликі сервери.

SQLite

Особливості: вбудована, файлова СУБД, не вимагає серверної установки; дуже легка (один файл бібліотеки). **Переваги:** надзвичайно проста інтеграція, чудова для мобільних/вбудованих додатків, тестування, прототипування; не потребує адміністрування. **Недоліки:** обмежена багатокористувацька конкурентність при записі; не підходить для великих розподілених систем; обмежені можливості масштабування.

Доречно використовується для: мобільних додатків, десктоп-програми, кешування, конфіг-файли.

Інші

- **Cassandra** — колоночна NoSQL СУБД для дуже великих розподілених навантажень і лінійної масштабованості (слабкіша консистентність за замовчуванням).
- **MongoDB** — документоорієнтована NoSQL (JSON-подібні документи), гнучка схема, хороша для швидкого прототипування і складних документних структур.
- **Redis** — in-memory key-value store, використовується як кеш або для швидких операцій; не є традиційною реляційною СУБД.
- **CockroachDB, YugabyteDB** — NewSQL проекти для глобально-розподілених транзакцій із сумісністю PostgreSQL у деяких випадках.

Порівняльна таблиця

СУБД	Архітектура	Розширюваність / Процедури	Переваги	Недоліки	Найкраще для
Access	Файлова (desktop)	Макроси, VBA	Швидкий старт, GUI	Погана масштабованість/безпека	Малі офісні додатки, прототипи
SQLite	Вбудована (файл)	Обмежені (тригери, функції)	Легка інтеграція, без адмінства	Конкурентність запису обмежена	Мобільні/вбудовані додатки, тест
MySQL (InnoDB)	Client-server	Stored procedures, triggers	Простота, велика екосистема	Деякі SQL-відхилення, складні аналітики	Веб/серверні додатки, LAMP
MariaDB	Client-server	розширені движки, процедури	Сумісний з MySQL + нові движки	Іноді сумісність із MySQL викликає питання	Замінник MySQL для веб/корп
PostgreSQL	Client-server	Потужні PL (PL/pgSQL, PL/Python), розширення	Стандарт-сумісність, розширюваність, надійність	Може потребувати тонкого тюнінгу	Складні системи, аналітика, GIS
MS SQL Server	Client-server	T-SQL, CLR, SSIS/SSRS/SSAS	Інтеграція з MS-екосистемою, BI	Вартість ліцензій	Корп. додатки, BI
Oracle DB	Client-server, RAC	PL/SQL, багаті функції	Максимальна корпоративна функціональність	Дуже висока вартість, складність	Критичні банки/ERP/великий дата-центр
Firebird	Файлова / Сервер	PSQL (вбудовані процедури)	Легка, стабільна, вбудована	Менша екосистема/інструменти	Десктоп/вбудовані системи, локальні сервери
MongoDB	Client-server	JavaScript-based	Гнучка схема, документоорієнтована	ACID для документів з обмеженнями; операції складніші	Документоорієнтовані додатки
Cassandra	Розподілена	CQL (обмежено SQL)	Масштабованість, доступність	Складність з транзакціями/консистентністю	Лог-тривалі дані, телеком, телеметрія
Redis	In-memory key-value	Lua скрипти	Швидкість, кешування, pub/sub	Обмеженість як повноцінної БД	Кеш, семафори, черги

При виборі СУБД зазвичай оцінюють наступні параметри

1. **Визначте тип навантаження:** OLTP (висока частота записів/транзакцій) — обирайте PostgreSQL, MS SQL, Oracle або

MySQL/InnoDB; аналітика/OLAP — PostgreSQL (з розширеннями) або спеціалізовані DW-системи.

2. **Масштаб і бюджет:** для обмеженого бюджету і необхідності відкритого коду — PostgreSQL або MariaDB/MySQL; для enterprise-level із підтримкою — Oracle або MS SQL (з бюджетом на ліцензії).

3. **Гнучкість схеми:** необхідна динамічна/документна структура — розгляньте MongoDB або гібридні підходи.

4. **Простота і вбудованість:** мобільні чи desktop-додатки — SQLite або Firebird (для десктопів).

5. **Гео-розподілені системи:** якщо потрібно багато-регіональне збереження з високою доступністю — NewSQL (CockroachDB, Yugabyte) або Cassandra залежно від вимог до консистентності.

§3 Архітектура баз даних

Основною ідеєю специфікації ANSI SPARC є виділення трьох архітектурних рівнів бази даних, а саме: *зовнішнього, концептуального* та *внутрішнього* (рис. 5).

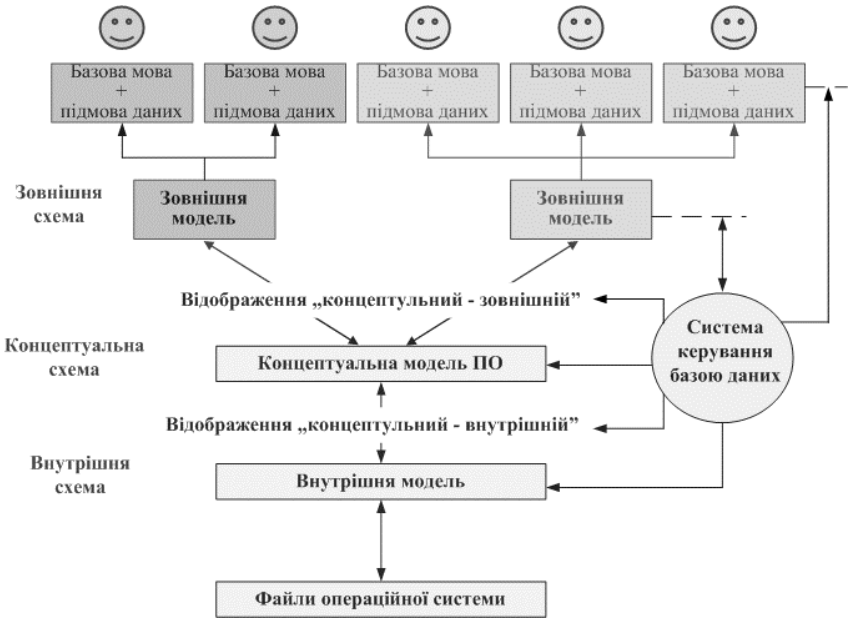


Рисунок 5.Трирівнева архітектура СКБД

3.1. Концептуальний рівень

На концептуальному рівні здійснюється інтегрований опис предметної області, для якої розробляється БД, незалежно від її сприйняття окремими користувачами та способів реалізації в комп’ютерній системі. Дамо означення основних понять, що використовуються на концептуальному рівні.

Предметна область (ПО) - частина реального світу, для якої здійснюється концептуальне моделювання.

Концептуальна модель ПО формальне зображення сукупності думок, які характеризують можливі стани ПО, а також переходи з одного стану в інший (включно з класифікацією наявних у ПО сутностей, чинних правил, законів, обмежень тощо)

Концептуальне моделювання ПО процес побудови концептуальної моделі ПО, яка б відображала ПО з урахуванням вимог, висунутих до цього процесу

Концептуальна схема - фіксація концептуальної моделі ПО засобами конкретних мов моделювання даних. У СКБД концептуальна модель подається у вигляді концептуальної схеми.

Опишемо властивості концептуальної моделі (схеми) й характерні особливості концептуального моделювання:

- Спільне та однозначне тлумачення предметної області всіма зацікавленими особами.
- Концептуальна схема відображує лише концептуально важливі аспекти ПО, виключаючи будь-які аспекти зовнішнього або внутрішнього відображення даних. Ця модель не повинна відображувати конкретні потреби окремих користувачів або додатків. Вона має фіксувати, чим є ПО в цілому, а не з точки зору інтересів або потреб користувачів.
- Визначення допустимих меж еволюції бази даних. У процесі експлуатації база даних може розвиватися, проте цей розвиток може відбуватися тільки в межах, допустимих для концептуальної схеми.
- Відображення зовнішніх схем на внутрішню. Саме через концептуальну схему зовнішні дані відображаються на внутрішні, й навпаки. В такий спосіб створюється єдина основа для опису даних і підтримки цих відображень.
- Забезпечення незалежності даних. Наявність відображень *концептуальний-зовнішній* і *концептуальний-внутрішній* дає змогу вирішувати проблему логічної та фізичної незалежності даних. Будь-які зміни в тій чи іншій зовнішній моделі не повинні спричиняти зміни в концептуальній або внутрішній моделях. У цьому випадку має змінитися тільки відповідне відображення «концептуальний-зовнішній». Аналогічно, будь-які зміни у внутрішній моделі не зачіпають концептуальну модель і моделі зовнішнього рівня, а тільки призводять до змін відображення «концептуальний-внутрішній».

- Централізоване адміністрування. Саме через концептуальну схему здійснюється адміністрування баз даних.
- Стійкість. Концептуальна схема не має підладжуватися до вимог тих чи інших користувачів (зовнішній рівень) або до вимог зберігання даних (внутрішній рівень). Будучи моделлю ПО, вона має змінюватися тільки тоді, коли входить у суперечність із нею.

3.2. Зовнішній рівень

Через зовнішній рівень користувачі та додатки отримують доступ до бази даних. Мета зовнішнього рівня - надати користувачу/додатку лише ті дані, які йому потрібні (а отже, до яких дозволений доступ) і в потрібному вигляді. Це індивідуальний рівень користувача, яким може бути кінцевий користувач, програміст чи додаток.

Зовнішня модель - це засоби зображення концептуальної моделі ПО з урахуванням інтересів конкретних користувачів або додатків. Кожна зовнішня модель подається в СКБД у вигляді зовнішньої схеми.

Зовнішній рівень виконує такі функції:

- Забезпечує зображення даних зручним для людини або додатку способом. Ступінь незалежності зовнішнього зображення від концептуального рівня визначається потужністю засобів опису відображення «концептуальний- зовнішній».
- Сприяє вирішенню проблеми безпеки (захисту) даних. Надаючи користувачу лише ті дані, що його цікавлять, ми залишаємо поза межами його доступу решту даних.
- Сприяє вирішенню проблеми логічної незалежності даних Це досягається завдяки відображенню «концептуальний-зовнішній», що встановлює відповідність між концептуальною схемою і конкретною зовнішньою схемою. Потужність його засобів визначає ступінь логічної незалежності додатків від даних.

3.3. Внутрішній рівень

Внутрішня модель є відображенням концептуальної моделі ПО з урахуванням способів зберігання даних і методів доступу до них. Внутрішня модель відображується в СКБД у вигляді внутрішньої схеми. Доступ до фізичної пам’яті надається за допомогою опису відображень внутрішньої моделі на фізичну пам’ять операційної системи.

Загалом внутрішній рівень виконує такі функції:

- Забезпечує налаштування бази даних для підвищення продуктивності обробки даних, опису й підтримки планованої надлишковості.
- Дає змогу описувати й підтримувати структури зберігання та методи доступу.
- Сприяє вирішенню проблеми фізичної незалежності даних: зміни у внутрішній схемі не повинні призводити до змін у зовнішній схемі.
- Сприяє вирішенню проблеми безпеки (захисту) даних.
- Вирішує проблему відображення даних на структури ОС, у яких дані зберігаються (до таких структур належать зокрема файли).

3.4. Моделі даних

3.4.1. Поняття моделі даних

Подання інформації за допомогою даних вимагає уніфікованого підходу до поняття даних як незалежного об'єкта моделювання. Тому для розробника ІС вибір відповідної моделі даних є однією з найважливіших проблем. Вибір моделі даних спричиняє вибір засобів аналізу ПО як області реального світу, що підлягає вивченню й обробці. Модель даних обмежує можливість вибору СКБД, тому що, як правило, окремо взята модель підтримує певну модель даних. Таким чином, поняття моделі даних є одним з фундаментальних понять

інформатики, від якого багато в чому залежать механізми реалізації ІС як програмно-апаратного комплексу.

Основою для будь-якої структури даних є відображення елементарної одиниці даних у вигляді такої трійки: <об’єкт, властивість об’єкта, значення властивості>. Сукупність взаємопов’язаних між собою елементарних одиниць даних може відображуватися різноманітними способами, що приводить до формування різних структур, а відтак - різних моделей даних. Моделі даних поділяються на два класи: сильно та слабко типізовані

У сильно типізованих моделях усі дані мають належати до певної категорії, або типу. Якщо дані не підпадають під жодну з категорій, їх потрібно типізувати штучно. Деякі моделі будуються у такий спосіб, що категорії визначаються наперед і не можуть змінюватися динамічно. У цьому випадку модельований світ начебто вміщується в гамівну сорочку. Наприклад, категорія «службовець» — строго фіксована, й усі її об’єкти повинні мати однакові властивості та структуру.

Сильно типізовані моделі мають значні переваги, бо дають змогу побудувати абстракції властивостей даних і дослідити їх у термінах категорій. Більшість моделей, що використовуються в автоматизованих системах, зокрема й базах даних, належать до сильно типізованих.

Для слабко типізованих моделей належність даних до тієї чи іншої категорії не має жодного значення. Категорії використовуються настільки, наскільки це доцільно в кожному конкретному випадку. Окремі дані можуть існувати як незалежно, так і у зв’язку з іншими. Інформація про категорії (якщо вони використовуються) розглядається як додаткова. На відміну від сильно типізованих моделей, слабко типізовані забезпечують інтеграцію даних і категорій. Найкращі можливості такої інтеграції надаються численням предикатів, яке у багатьох моделях даних використовується для зображення знань, що не підтримуються базовими засобами моделювання.

Модель даних (Data Model) є логічною структурою даних, що становить притаманні цим даним властивості, незалежні від апаратного й програмного забезпечення й не пов'язані з функціонуванням комп'ютера.

Можна розглянути кілька аспектів моделювання в обробці даних:

- інформаційне моделювання:
- концептуальне моделювання (моделювання семантики ПО);
- логічне моделювання даних;
- фізичне моделювання:
- створення моделей доступу до даних;
- оптимізація фізичної організації даних в апаратному середовищі.

На рис. 6 ілюструється загальний зміст поняття моделі даних на цей час.

Рисунок 6.Подання про інформаційну модель даних

3.4.2. Основні типи моделей та їх еквівалентність

Наявність у СКБД певної структури даних приводить до поняття баз структурованих даних, тобто дані в таких БД повинні бути представлені як сукупність взаємозалежних елементів. Слід мати на увазі, що для кожного типу БД використовуються відповідні моделі даних.

У цей час для баз структурованих даних розрізняють три основні типи логічних моделей даних залежно від характеру підтримуваних ними зв'язків між елементами даних - мережну, ієрархічну й реляційну.

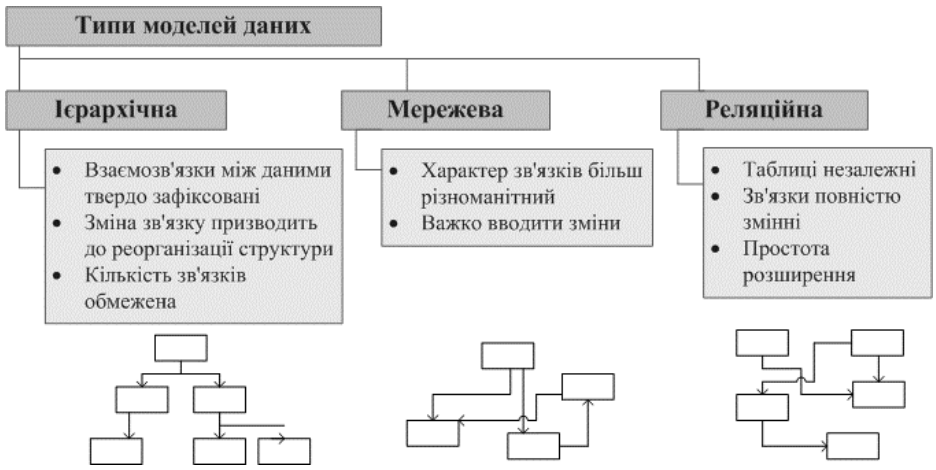


Рис.7. Особливості моделей даних

Рис. 7 ілюструє особливості кожної моделі даних. При зіставленні моделей варто пам'ятати, що всі вони теоретично еквівалентні. Еквівалентність моделей полягає в тому, що вони можуть бути зведені одна до одної шляхом формальних перетворень.

3.5. Основи реляційної моделі даних

Основою сучасної теорії баз даних є реляційна модель. Основоположником теорії реляційних баз даних вважається співробітник фірми IBM доктор Е.Кодд, який опублікував у 1970 р. статтю „Реляційна модель даних для великих колективних банків даних.” Е. Кодд, будучи математиком за освітою, запропонував використати для обробки даних апарат теорії множин (об’єднання, декартів добуток, різницю) і логіки. Він довів, що довільний набір даних можна подати у вигляді особливих двомірних таблиць, відомих у математиці як „відношення” (relation) – звідки і пішла назва “реляційні бази даних”. Теорія реляційних баз даних, розроблена Е. Коддом, має під собою потужну математичну основу, яка описує правила ефективної організації даних. У реляційній моделі розглядаються три аспекти даних: структура даних (об’єкти даних), цілісність даних і обробка даних.

3.5.1. Реляційні об’єкти даних

Розглянемо спочатку найважливіші терміни, які використовуються в частині об’єктів: це відношення, кортеж, кардинальне число, атрибут, степінь, домен, первинний ключ.

Формальний реляційний термін	Неформальний еквівалент
Відношення	Таблиця
Кортеж	Рядок або запис
Атрибут	Стовпець або поле
Кардинальне число	Кількість рядків
Степінь	Кількість стовпців

Відношення відповідає тому, що ми до цього часу називали таблицею. ***Кортеж*** відповідає рядку таблиці. Кількість кортежів називається ***кардинальним числом***, а кількість атрибутів - ***степенем***.

Домен – це загальна сукупність значень, з якої беруться значення для певних атрибутів даного відношення.

Розглянемо приклад відношення **Постачальник**.

Код	Ім'я	Статус	Місто
301	Іваненко	20	Київ
302	Шевчук	10	Львів
303	Сидорюк	20	Львів
304	Петренко	30	Київ
305	Гринів	10	Тернопіль

У цьому відношенні: степінь – 4, кардинальне число – 5, домени: для атрибуту **Статус**: 10,20,30; для атрибуту **Місто**: Київ, Тернопіль, Львів.

Домен – це іменована множина скалярних значень одного типу. Кожний атрибут визначений на єдиному домені.

Визначення відношення. Відношення визначається на множині доменів і складається з двох частин: заголовка і тіла. ***Заголовок*** – це рядок заголовків стовпців. ***Тіло*** – множина рядків даних. Тіло містить множину кортежів. Кортеж містить множину пар: ім'я атрибута і значення атрибута.

Відношення має такі властивості:

- немає однакових кортежів;
- кортежі не впорядковані зверху вниз;

- атрибути не впорядковані зліва направо;
- всі значення атрибутів атомарні.

Перша властивість випливає з того, що відношення – це множина кортежів, а множина складається з різних елементів. Ця властивість є ілюстрацією того, що відношення і таблиця не одне й те ж, оскільки таблиця може містити однакові рядки.

Друга і третя властивості випливають з того, що кортежі і атрибути – це множини, а множини загалом невпорядковані, немає поняття першого, другого атрибута чи кортежа. Це відрізняє таблицю від відношення.

Четверта властивість означає, що всі значення атрибутів атомарні – неподільні. Відношення, яке задовольняє цю умову, називається ***нормалізованим***, або представленим в 1-й нормальній формі (1НФ), є ще 2НФ, 3НФ, 4НФ та ін.

Отже, відношення з точки зору реляційної моделі завжди означає нормалізоване відношення. Нормалізовані відношення простіші для обробки (легше вставити чи вилучити запис).

Реляційна теорія забороняє використання неатомарних значень атрибутів. Але в реальному житті бувають протилежні випадки. Наприклад, в історичних записах іноді зустрічаються такі: „дата народження невідома” або в паспортному столі „без певного місця проживання”. Для подолання цієї проблеми Е.Кодд запропонував для наведення невідомої чи відсутньої інформації використовувати спеціальні маркери, які називаються NULL – значеннями. Тобто, якщо зустрічається в позиції атрибута значення NULL – це означає, що воно відсутнє. Звернемо увагу, що NULL – це маркер, а не порожня множина.

3.5.2. Поняття ключа та типи ключів. Цілісність даних

Одним із основних понять реляційної теорії баз даних є поняття ключа. Будь-яке відношення містить один або кілька атрибутів, значення яких однозначно ідентифікують кожний його кортеж.

Ключі бувають таких типів: потенційний, первинний, альтернативний, зовнішній.

Потенційний (можливий) ключ має кожне відношення (у всякому разі один). Потенційний ключ – це той можливий ключ, за яким можна знайти кортеж.

Визначення потенційного ключа. Нехай задано деяке відношення R. Тоді потенційний ключ К – це підмножина атрибутів відношення R, яка має такі властивості:

- *унікальність* (немає різних кортежів у відношення R з однаковим значенням К);
- *ненадлишковість* – жодна з підмножин К не володіє властивістю унікальності. Наприклад, не варто створювати ключ, який містить і табельний номер, і номер паспорта, досить вибрати будь-який із них, щоб знайти запис.

Звернемо увагу, що кожне відношення має щонайменше один потенційний ключ, оскільки не містить однакових кортежів, тобто оскільки кортежі унікальні, то комбінація всіх атрибутів має властивість унікальності.

Первинні та альтернативні ключі. Відношення може мати більше одного потенційного ключа. В такому випадку в реляційній моделі один із потенційних ключів вибирають як первинний, а інші потенційні ключі називають альтернативними. Такий вибір є довільним. Якщо потенційний ключ один, то він і є первинним.

Зовнішні ключі. Розглянемо базу даних **Постачальники деталей**. Вона складається з трьох таблиць (відношень): **Постачальник** (код_п, ім’я, місто); **Деталь**(код_д, назва, кількість); **Постачальники_деталей** (код_п, код_д, кількість, дата)

Розглянемо атрибут **Код_п** відношення **Постачальник**. Ясно, що значення для цього атрибута повинно бути допустимим лише тоді, коли таке значення існує як первинний ключ **Код_п** відношення **Постачальник**. Якщо немає постачальника, то немає кому постачати деталь. Аналогічно **Код_д** у **Постачальник_деталей** і **Код_д** у відношення **Деталь**. Якщо деталь не вироблена, то вона не може бути поставлена.

Тут **Код_п** і **Код_д** відношення **Постачальники_деталей** є зовнішніми ключами.

Визначення зовнішнього ключа. Нехай R2 – відношення, FK – зовнішній ключ відношення R2. Зовнішній ключ FK – це підмножина атрибутів з R2 така, що:

- існує цільове відношення R1 з потенційним ключем СК.
- кожне значення зовнішнього ключа FK відношення R2 збігається зі значенням СК відношення R1. Тобто кожне значення зовнішнього ключа повинно бути значенням відповідного потенційного ключа.

Якщо зовнішній ключ простий, то і потенційний ключ простий. Якщо зовнішній ключ складений, то і потенційний ключ складений. Зовнішній ключ не повинен бути первинним для цього відношення. В даному прикладі обидва ключі складають первинний.

Поряд із поняттям зовнішнього ключа реляційна модель включає ***правило цілісності***. База даних не повинна містити неузгоджених значень зовнішніх ключів, тобто таких значень зовнішнього ключа, для яких не існує відповідного йому значення потенційного ключа в цільовому відношенні.

База даних повинна бути в такому стані, щоб не порушувалось правило посилальної цілісності.

3.5.3. Функціональні залежності

Одним із аспектів теорії баз даних є нормалізація відношень. Вона ґрунтується на тій чи іншій залежності між атрибутами відношення. Визначимо основні типи залежностей.

Нехай R – відношення, X і Y – довільні підмножини множини атрибутів. Тоді Y ***функціонально залежить*** від X, X->Y (X функціонально визначає Y), коли кожне значення X пов'язано точно з одним значенням Y. Інакше: якщо два кортежі збігаються за значенням X, то вони збігаються за значенням Y.

Якщо X – потенційний ключ, то всі атрибути функціонально залежать від нього.

Якщо у відношенні R: A->B і A не є потенційним ключем, то R характеризується надлишковістю. A->B виконується тоді й тільки тоді, коли два кортежі з однаковими значеннями A мають однакові значення B. Наприклад tab->adres.

У випадку наявності в таблиці складеного ключа використовується поняття ***повної функціональної залежності***.

Повною функціональною залежністю неключових атрибутів називається така залежність, при якій кожний неключовий атрибут функціонально залежить від усього ключа (тобто від усієї сукупності атрибутів ключа), але не знаходиться у функціональній залежності від жодної частини складеного ключа. Наприклад fio+data->adres.

Транзитивна залежність спостерігається в тому випадку, якщо один із двох неключових атрибутів залежить від ключа, а другий неключовий атрибут залежить від першого. A->B->C (C транзитивно залежить від A)

Поняття ***нетранзитивної*** залежності протилежне поняттю функціональної залежності. Тобто нетранзитивна залежність спостерігається в тому випадку, якщо жоден із неключових атрибутів функціонально не залежить від довільного іншого неключового атрибута.

Багатозначна залежність. Атрибут А багатозначно визначає атрибут В, якщо для кожного значення атрибута А існує добре визначена множина відповідних значень атрибута В.

3.6. Проектування бази даних

При проектуванні бази даних замовник, як правило, подає розробнику опис форм, бланків, які існують у паперовому вигляді. При простому перенесенні полів паперових форм у таблиці бази даних неминуче виникає низка проблем. При проектуванні баз даних необхідно дотримуватись таких правил:

- виключити надлишковість збережуваних даних;
- звести кількість таблиць до мінімуму.

Можна сказати, що основною метою проектування баз даних є виключення надлишковості збережуваних даних. Це приведе до економії обсягу пам’яті, усуненню неузгодженості (суперечностей) відомостей про один і той же об’єкт, які зберігаються в різних місцях. Крім того, при зміні якихось відомостей про об’єкт нам не потрібно буде змінювати копії цих відомостей, які зберігаються в іншому місці. Цих копій просто не буде.

3.6.1. Нормалізація та її необхідність

Проілюструємо хід проектування бази даних на прикладі магазину комп’ютерної техніки. Всі дані занесемо в таблицю 1.

Таблиця 1. Магазин „КТ”

Дата	Ім’я продавця	Ім’я покупця	Адреса	Назва товару	Ціна
1.02.25	Тепляк Олексій	Марчук Сергій	Чернівці, вул. Головна 129/54	Комп’ютер	450
8.02.25	Поштар Андрій	Данилів Євген	Чернівці, вул. Шевченка 34	Принтер Модем	94
8.02.25	Савчук Сергій	Ємчук Микола	Кіцмань, вул. Л. Українки 10,к.1	Мишка	
15.02.25		Мазур Василь		Монітор	129

22.02.25 22.02.25	Кушнір Анатолій Поштар Андрій Савчук Сергій	Гульчак Іван Прошенко Вікторія	Хотин, вул. Хоткевича 54, кв.2 Чернівці, вул. Лисенка 8,кв.5 Чернівці, вул. Авангардна, 24 /12	Клавіатура Клавіатура Принтер Монітор	 270 496

Звичайно, таблицю 1 відношенням назвати не можна, оскільки деякі її стовпці Ім’я продавця, Ім’я покупця, Адреса, Назва товару містять неатомарні значення. Для усунення цього недоліку додамо до таблиці стовпці Прізвище покупця, Прізвище продавця. Стовпець Адреса розіб’ємо на два стовпці: Місто, вулиця. Це дозволить виконати статистичну обробку купівельної здатності міст і їх окремих районів. Крім того, до таблиці доведеться додати кілька рядків, щоб значення стовпця Назва товару також стали атомарними.

Таблиця 2. Магазин „КТ”

Дата	**Прізвище продавця**	**Ім’я продавця**	**Прізвище покупця**	**Ім’я покупця**	**Місто**	**Вулиця**	**Назва товару**	**Сума**
1.02.25	Тепляк	Олексій	Мальчук	Сергій	Чернівці	Головна	Комп’ютер	450
8.02.25	Поштар	Андрій	Данилів	Євген	Чернівці	Шевченка	Принтер	76
8.02.25	Поштар	Андрій	Данилів	Євген	Чернівці	Шевченка	Модем	18
8.02.25	Савчук	Сергій	Ємчук	Микола	Кіцмань	Л.Українки	Миша	12
8.02.25	Савчук	Сергій	Ємчук	Микола	Кіцмань	Л.Українки	Монітор	120
8.02.25	Савчук	Сергій	Ємчук	Микола	Кіцмань	Л.Українки	Клавіатура	17
15.02.25	Кушнір	Анатолій	Мазур	Василь	Хотин	Хоткевича	Клавіатура	17
22.02.25	Поштар	Андрій	Гульчак	Іван	Чернівці	Лисенка	Принтер	76
22.02.25	Поштар	Андрій	Гульчак	Іван	Чернівці	Лисенка	Монітор	120

Таблиці, подібні наведеній вище, називаються універсальними таблицями баз даних. Вони придатні для збереження всієї інформації, яку

потрібно внести до бази даних. Для простоти припускається, що назви вулиць у кожному місті унікальні. Припускається, що кількість полів в універсальних таблицях не перевищує 15.

Таке подання даних у вигляді таблиці 2 приводить до виникнення великого обсягу надлишкових даних. Ім'я і адресу одного й того ж покупця необхідно заново вносити при введенні нового замовлення. Є інші проблеми: проблема великої кількості помилок при введенні великої кількості інформації, яка повторюється.

Таким чином, при використанні універсальних таблиць виникають такі проблеми:

• Проблема надлишковості даних. Дані практично всіх стовпців багатократно повторюються.

• Проблема оновлення баз даних (або потенційної суперечності даних). Внаслідок надлишковості можна обновити, наприклад, адресу покупця в одному рядку, залишаючи її незмінною в іншому. З'явиться дві адреси, одна з яких вже недійсна.

• Проблема додавання даних. Додавання нових даних потребує введення даних для всіх стовпців, навіть, якщо в стовпці вже необхідні дані є. Очевидна надлишковість даних. Рано чи пізно при введенні даних, які дублюються, буде допущена помилка. Це приведе до виникнення двох різних значень. Крім того, неможливо додати інформацію раніше, ніж вона буде потрібна. Стосовно нашої таблиці це означає, що не можна ввести ім'я продавця, доки він не продасть якийсь товар.

• Проблема вилучення даних. При вилученні даних одного рядка може бути загублена якась потрібна інформація. Для усунення описаних проблем необхідно використати так звану нормалізацію.

Нормалізація – це формальний апарат обмежень на формування таблиць, який описує розбиття таблиць на дві або більше таблиць. Кінцевою метою нормалізації є одержання такого проекту БД, в якому будь-яка частина інформації зберігається в одному місці, тобто виключається надлишковість

інформації. Це робиться не стільки з метою економії місця (в деяких випадках нормалізовані таблиці займають більше місця, ніж ненормалізовані), скільки для виключення можливості суперечностей збережуваних даних.

Отже, Таблиця 2 є прикладом ненормалізованої таблиці.

Основою процесу нормалізації є запропонований Е. Коддом процес нормалізації відношень. Процес нормалізації – це послідовна зміна структури таблиць доти, доки вони не будуть задовольняти вимоги останньої форми нормалізації. Існують такі шість форм нормалізації:

- Перша нормальна форма – 1НФ
- Друга нормальна форма – 2НФ
- Третя нормальна форма – 3НФ
- Нормальна форма Бойса-Кодда – НФБК
- Четверта нормальна форма – 4НФ
- П’ята нормальна форма – 5НФ.

На практиці найчастіше використовуються перші три нормальні форми, які забезпечують достатній рівень нормалізації і водночас легко реалізуються на практиці. При зведенні до останньої, п’ятої нормальної форми таблиця вважається повністю нормалізованою.

3.6.2. Перша нормальна форма

Таблиця знаходиться в першій нормальній формі тоді й тільки тоді, коли вона не містить однакових полів і складених значень полів, тобто поля не повторюються, значення в них атомарні. Отже, Таблиця 1 не знаходиться в 1НФ, а Таблиця 2 знаходиться в 1НФ. Вище зазначалося, що будь-яке відношення за його визначенням знаходиться в першій нормальній формі.

3.6.3. Друга нормальна форма

Таблиця знаходиться в другій нормальній формі, якщо вона задовольняє вимоги першої нормальної форми, і всі її поля, які не входять до первинного ключа, пов’язані повною функціональною залежністю з первинним ключем,

тобто будь-яке неключове поле однозначно ідентифікується повним набором ключових полів.

Іншими словами таблиця, яка знаходиться в другій нормальній формі, повинна відповідати таким вимогам:

- Таблиця повинна мати дані про один тип об’єктів.
- Кожна таблиця повинна містити одне або кілька полів, які утворюють унікальний ідентифікатор (первинний ключ) для кожного рядка даної таблиці.
- Всі поля, які не входять до первинного ключа, повинні однозначно визначатись первинним ключем (залежати від нього).

Очевидно, що в Таблиці 2 немає первинного ключа. Місто залежить від покупця. Товар залежить від дати і покупця.

Тому розіб’ємо Таблицю 2 на такі 5 таблиць: Покупець, Продавець, Товар, Замовлення, Замовлення товару.

Просте розбиття таблиці на кілька дрібніших таблиць може привести до втрати логічного зв’язку між даними цих таблиць. При цьому необхідно передбачити створення допоміжних стовпців. Нові стовпці можуть бути використані як первинний ключ у кожній таблиці. Використовуючи цей ключ, можна буде легко зв’язувати дані, розміщені в різних таблицях.

Таблиця 2.1. Продавець

ID продавця	Ім’я продавця	Прізвище продавця
1	Олексій	Тепляк
2	Андрій	Поштар
3	Сергій	Савчук
4	Анатолій	Кушнір

Таблиця 2.2.Покупець

ID покупця	Ім’я покупця	Прізвище покупця	Місто	Вулиця
1	Сергій	Марчук	Чернівці	Головна
2	Євген	Данилів	Чернівці	Шевченка
3	Микола	Ємчук	Кіцмань	Л. Українки
4	Василь	Мазур	Хотин	Хоткевича
5	Іван	Гульчак	Чернівці	Лисенка

Таблиц 2.3. Товар

ID товару	Назва товару	Ціна
1	Комп’ютер	450
2	Принтер	76
3	Клавіатура	7
4	Монітор	120
5	Мишка	2
6	Модем	18

Таблиця 2.4. Замовлення

ID замовлення	Дата	ID продавця	ID покупця	Сума покупки
1	1.02.25	1	1	450
2	8.02.25	2	2	94
3	8.02.25	3	3	129
4	15.02.25	4	4	7
5	22.02.25	2	5	196

Таблиця 2.5. Замовлення товару

ID замовлення	ID товару
1	1
2	2
2	6

3	5
3	1
3	3
4	3
5	2
5	4

Зведення таблиці до другої нормальної форми дозволяє уникнути повторення (надлишковості) одних і тих же даних, які з’явились після зведення до першої нормальної форми.

3.6.4. Третя нормальна форма

Таблиця знаходиться в третій нормальній формі, якщо вона відповідає визначенню другої нормальної форми, і жодне з її неключових полів функціонально не залежить від будь-якого іншого неключового поля. Або, іншими словами, таблиця знаходиться в 3НФ, якщо вона знаходиться в 2НФ, і кожне неключове поле транзитивно не залежить від первинного ключа. Вимога 3НФ полягає в тому, щоб усі неключові поля залежать тільки від первинного ключа і не залежать один від одного. Щоб була можливість змінювати значення будь-якого неключового поля, не змінюючи значення будь-якого іншого поля таблиці.

Із усіх таблиць, які були зведені до другої нормальної форми, тільки Таблиця 2.2 не відповідає визначенню 3НФ. Справа в тому, що поле Вулиця залежить від поля Місто (вулиці вважаються унікальними в містах).

Додамо ще такі таблиці.

Таблиця 2.6. Список міст

ID міста	Місто
1	Чернівці
2	Кіцмань
3	Хотин

Таблиця 2.7. Адреси

ID адреси	ID міста	Вулиця
1	1	Головна
2	1	Шевченка
3	2	Л.Українки
4	3	Хоткевича
5	1	Лисенка

Тоді Таблицю 2.2 треба змінити, замінивши поля Місто та Вулиця одним полем ID адреси. В результаті Таблиця 2.2 буде мати такий вигляд:

Таблиця 2.7

ID покупця	Ім’я покупця	Прізвище покупця	ID адреси
1	Сергій	Марчук	1
2	Євген	Данилів	2
3	Микола	Ємчук	3
4	Василь	Мазур	4
5	Іван	Гульчак	5

3.6.5. Нормальна форма Бойса-Кодда

При зведенні відношень за допомогою алгоритму нормалізації до відношень у ЗНФ неявно припускається, що всі відношення містять один потенційний ключ. Це не завжди правильно, бо у відношення може бути кілька потенційних ключів. Точніше, визначення Кодда для третьої нормальної форми не зовсім підходить у випадку, якщо, по-перше, у відношеннях існує два або декілька потенційних ключів, по-друге, ці потенційні ключі є складеними, і, по-третє, мають принаймні спільний один із одним атрибут. Тому це визначення перероблене й дістало назву третьої нормальної форми Бойса-Кодда, оскільки було сформульовано ними обома.

Відношення знаходиться в нормальній формі Бойса-Кодда тільки в тому випадку, якщо довільна функціональна залежність між його атрибутами зводиться до повної функціональної залежності від можливого ключа.

Будь-яке відношення в нормальній формі Бойса-Кодда є відношенням у ЗНФ. Навпаки не правильно.

3.6.6. Четверта нормальна форма

Визначення четвертої нормальної форми засноване на визначенні багатозначної залежності.

Нехай дано не нормалізоване відношення УКВП (тобто відношення, яке не знаходиться в 1НФ), яке містить інформацію про курси, викладачів та підручники

Таблиця 3. УКВП

Курс	Викладач	Підручник
Фізика	Іваненко Ткаченко	Механіка Оптика
Математика	Іваненко	Математичний аналіз Алгебра Аналітична геометрія

Перетворимо дане відношення в еквівалентне нормалізоване

Таблиця 3.1. К_В_П

Курс	Викладач	Підручник
Фізика	Іваненко	Механіка
Фізика	Іваненко	Оптика
Фізика	Ткаченко	Механіка
Фізика	Ткаченко	Оптика
Математика	Іваненко	Математичний аналіз
Математика	Іваненко	Алгебра
Математика	Іваненко	Аналітична геометрія

Очевидно, що відношення КВП означає, що кортеж Курс : c, Викладач : t, Підручник : x з’являється в даному відношенні тоді й тільки тоді, коли курс

c читається викладачем **t** з використанням підручника x. Тоді, беручи до уваги те, що для даного відношення існують всі можливі комбінації викладачів разом із підручниками, можна стверджувати, що для відношення КВП виконується така умова: якщо присутні два кортежі (c, t1, x1), (c, t2, x2), тоді присутні кортежі (c, t1, x2), (c, t2, x1).

Очевидно, що відношення КВП характеризується значною надлишковістю і приводить до виникнення аномалій оновлення. Наприклад, для додавання інформації про те, що курс фізики може читатися новим викладачем, необхідно створити два нових кортежі, по одному до кожного підручника. Тим не менше відношення КВП знаходиться у НФБК, оскільки є "повністю ключовим". Подібні проблеми із відношеннями у НФБК виникають дуже часто. Інтуїтивно зрозуміло, що у відношенні КВП ці труднощі викликані тим, що викладачі та підручники цілком незалежні один від одного. Крім того, легко помітити, що ситуація може бути виправлена, якщо замінити відношення КВП двома – КВ (курс, викладач) і КП (курс, підручник).

Таблиця 3.2. К_В

Курс	Викладач
Фізика	Іваненко
Фізика	Ткаченко
Математика	Іваненко

Таблиця 3.3. К_П

Курс	Підручник
Фізика	Механіка
Фізика	Оптика
Математика	Математичний аналіз
Математика	Алгебра
Математика	Аналітична геометрія

У відношенні К_В_П є дві багатозначні залежності: курс –>> викладач (один курс читає багато викладачів) та курс –>> підручник (для одного курсу є багато підручників). Тут використано позначення A –>> B – B багатозначно залежить від A або A багатозначно визначає B.

Визначення багатозначної залежності. Нехай A, B, C – довільні підмножини множини атрибутів відношення R. Тоді B багатозначно залежить від A (A –>> B) тоді й тільки тоді, коли множина значень B, що відповідає парі

(A; C) відношення R, залежить тільки від A і не залежить від C. Можна довести, що для даного відношення R{A, B, C} багатозначна залежність A –>> B виконується тоді й тільки тоді, коли виконується багатозначна залежність A –>> C. Отже, багатозначні залежності завжди утворюють зв'язані пари A –>> B□C.

Таблиці К_В і К_П не містять багатозначних залежностей, тому є вдосконаленням структури КВП. Було б бажано відношення КВП замінити двома проекціями К_В і К_П.

Справедлива така теорема [1].

Теорема Фейгеня. Нехай A, B і C є множинами атрибутів R{A, B, C}. Відношення R буде дорівнювати з'єднанню його проекцій {A, B} і {A, C} тоді й тільки тоді, коли для відношення R виконується багатозначна залежність A –>> B|C.

Очевидно, що відношення К_В і К_П знаходяться у 4НФ.

3.6.7. П'ята нормальна форма

Для визначення 5НФ дамо визначення повної декомпозиції відношень.

Визначення. Повною декомпозицією (розбиттям) відношення називають таку сукупність довільного числа його проекцій, з'єднання яких повністю збігається із вмістом вихідного відношення.

Проекція – це копія відношення, яка не містить один або кілька атрибутів вихідного відношення. Кожна проекція складається з одного або кількох потенційних ключів і може містити додаткові атрибути.

Відношення в 5НФ гарантує, що воно не має аномалій, які можуть бути виключені розбиттям на проекції, загалом воно може містити аномалії.

Розглянемо відношення SPJ

Таблиця 4. SPJ

s	p	j

s_1	p_1	j_1
s_1	p_1	j_2
s_1	p_2	j_1
s_2	p_1	j_1

Відношення SPJ не знаходиться в 5НФ, оскільки воно може бути декомпоновано на 3 відношення.

Таблиця 4.1. SP

s	P
s_1	p_1
s_1	p_2
s_2	p_1

Таблиця 4.2. PJ

p	j
p_1	j_2
p_2	j_1
p_1	j_1

Таблиця 4.3. JS

j	s
j_2	s_1
j_1	s_1
j_1	s_2

Відношення SP, PJ, JS знаходяться в 5НФ, оскільки їх не можна декомпонувати.

Відношення в 5НФ можна декомпонувати, але кожна з проекцій у такому випадку буде містити один або кілька потенційних ключів.

3.7. Основні правила, які використовуються в процесі нормалізації

1. Відношення в 1НФ потрібно розбити на проекції для виключення всіх функціональних залежностей. У результаті одержується набір відношень у 2НФ.

2. Відношення у 2НФ розбивають на проекції для виключення будь-яких транзитивних функціональних залежностей. На цьому етапі одержується набір відношень у 3НФ.

3. Відношення в 3НФ треба розбити на проекції для виключення будь-яких функціональних залежностей, що залишилися, в яких детермінанта не є потенційними ключами. В результаті одержимо набір відношень у НФБК.

4. Відношення в НФБК необхідно розбити на проекції для виключення будь-яких багатозначних залежностей, які не є функціональними залежностями. В результаті буде одержаний набір відношень у 4НФ.

5. Відношення в 4НФ потрібно розбити на проекції для виключення будь-яких залежностей з'єднання, які не є потенційними ключами. Буде одержано набір відношень у 5НФ.

3.8. Процес проектування баз даних

1-ий етап. Обстеження предметної області. Це відбувається в результаті бесід із замовником. Потрібно вияснити:

- Які межі предметної області? Можливості її зміни й розвитку.
- Який список фрагментів предметної області?
- Одержати дані про кожний фрагмент.
- Які процеси передачі й обробки даних проходять у кожному фрагменті?
- Яка існує технологія накопичення й обробки інформації в предметній області?
- Як часто поступає і коректується інформація? Хто несе відповідальність за її достовірність?

2-ий етап.

- Знайомство з усіма документами, які циркулюють у кожному фрагменті.
- Визначення об'єктів та їх атрибутів.
- Визначення ключів для об'єктів, кодування об'єктів.
- Проектування і нормалізація відношень.

3-ій етап. Визначення запитів.

4-ий етап. Перевірка повноти й коректності проекту (це перевірка всіх запитів до БД)

§4. Основи реляційної алгебри

Крім структури даних, до складу *реляційної моделі даних,* входять операції маніпулювання даними. З усіх таких операцій складається *мова запитів.* Найвідомішими мовами запитів у реляційній моделі є реляційна алгебра та реляційне числення. Розглянемо мову реляційної алгебри її основні операції та їх характерні властивості.

У класичному розумінні *алгебра* визначається як пара, що складається з основної множини і множини операцій, при цьому аргументи й результат кожної операції належать основній множині.

Реляційна алгебра *є* алгеброю в строгому класичному розумінні її визначення. Елементами основної множини є реляційні відношення. У зв'язку з цим операції алгебри можуть вкладатися одна в одну, тобто аргументом певної операції може бути результат виконання іншої операції. Це дає можливість записувати запити довільного рівня складності у вигляді виразів, що містять вкладені одна в одну операції.

4.1. Основні поняття та означення

4.1.1. Реляційні відношення

Елементами основної множини в реляційній алгебрі є реляційні відношення. Реляційне відношення – це скінченна множина кортежів, що мають однакову схему (набір атрибутів). Атрибут – це іменована характеристика об'єкта, а домен – множина допустимих значень для атрибута.

4.1.2. Сумісність відношень

Два реляційні відношення $R_1(A_1, \dots, A_n)$ і $R_2(B_1, \dots, B_k)$ називаються *сумісними*, якщо:

1. У них однакова кількість атрибутів, тобто $k = n$.
2. Можна встановити взаємно однозначну відповідність між доменами атрибутів першої та другої реляцій. Формально, існує бієктивне

відображення $S: \{1, \dots, k\} \to \{1, \dots, k\}$ таке, що $Dom(A_i) = Dom(B_{S(i)})$ для $i = 1, \dots, k$.

Ця концепція є критичною для коректного виконання теоретико-множинних операцій, таких як об'єднання, перетин та різниця.

4.2. Теоретико-множинні операції

Ці операції вимагають сумісності операндів.

4.2.1. Операція об'єднання (Union)

Означення: Об'єднанням сумісних реляційних відношень R_1 і R_2 зі схемою L (позначається $R_1 \cup R_2$) називається відношення R зі схемою $R(L)$, що містить усі кортежі, які належать хоча б одному з відношень R_1 або R_2, без повторень.

$$R(L) = R_1(L) \cup R_2(L) = \{r \mid r \in R_1 \vee r \in R_2\}$$

Властивості:

- Комутативність: $R_1 \cup R_2 = R_2 \cup R_1$
- Асоціативність: $(R_1 \cup R_2) \cup R_3 = R_1 \cup (R_2 \cup R_3)$
- Дистрибутивність щодо перетину**:** $R_1 \cup (R_2 \cap R_3) = (R_1 \cup R_2) \cap (R_1 \cup R_3)$

Приклад:

Нехай:

- R_1: { (a_1, b_1), (a_1, b_2), (a_2, b_3) }
- R_2: { (a_1, b_1), (a_2, b_1) }

Тоді $R_1 \cup R_2$: { (a_1, b_1), (a_1, b_2), (a_2, b_3), (a_2, b_1) }

4.2.2. Операція перетину (Intersection)

Означення: Перетином сумісних реляційних відношень R_1 і R_2 (позначається $R_1 \cap R_2$) називається відношення R, що містить кортежі, які одночасно належать і R_1, і R_2.

$$R(L) = R_1(L) \cap R_2(L) = \{r \mid r \in R_1 \wedge r \in R_2\}$$

Властивості: Комутативна, асоціативна, дистрибутивна щодо об'єднання.

Приклад:

Для R_1 і R_2 з попереднього прикладу:

$$R_1 \cap R_2: \{ (a_1, b_1) \}$$

4.2.3. Операція різниці (Difference)

Означення: Різницею сумісних реляційних відношень R_1 і R_2 (позначається $R_1 - R_2$) називається відношення R, що містить ті кортежі з R_1, яких немає в R_2.

$$R(L) = R_1(L) - R_2(L) = \{r \mid r \in R_1 \wedge r \notin R_2\}$$

Властивості: Не комутативна, не асоціативна, не дистрибутивна.

Приклад:

Для R_1 і R_2:

$$R_1 - R_2: \{ (a_1, b_2), (a_2, b_3) \}$$

4.3: Спеціальні реляційні операції

Ці операції є унікальними для реляційної моделі.

4.3.1. Операція проекції (Projection)

Означення: Проекцією реляційного відношення $R(A_1, \dots, A_k)$ за атрибутами $A_{i1}, \dots, A_{in}$ (позначається $R[A_{i1}, \dots, A_{in}]$ або $\Pi_{A_{i1},\dots,A_{in}}(R)$) називається відношення, кортежі якого отримані з кортежів відношення R шляхом видалення значень, що не належать обраним атрибутам. Повторювані кортежі видаляються.

$$S = \Pi_{A_{i1},\dots,A_{in}}(R) = \{r[A_{i1}, \dots, A_{in}] \mid r \in R\}$$

Приклад:

Нехай R:

A	**B**	**C**
a_1	b_1	c_1
a_1	b_2	c_1

A	B	C
a_2	b_3	c_1
a_2	b_4	c_2

Тоді $\Pi_{A,C}(R)$:

A	C
a_1	c_1
a_2	c_1
a_2	c_2

4.3.2. Операція обмеження (селекції) (Selection)

Означення: Обмеженням (або селекцією) реляційного відношення R за умовою F (позначається $\sigma_F(R)$) називається відношення, що містить ті кортежі з R, які задовольняють умові F. Умова F будується з операндів-атрибутів, операторів порівняння $(=, <, >, \leq, \geq, \neq)$ та логічних зв'язок $(\wedge, \vee, \neg)$.

A	B	C
a_2	b_3	c_1

Приклад: Для R з попереднього прикладу: $\sigma_{A=a_2}(R)$:

a_2	b_4	c_2

4.3.3. Операція декартового добутку (Cartesian Product)

Означення: Декартовим добутком реляційних відношень $R(A_1, \dots, A_n)$ та $S(B_1, \dots, B_m)$ (позначається $R \times S$) називається відношення Q зі схемою $Q(A_1, \dots, A_n, B_1, \dots, B_m)$, яке містить усі можливі з'єднання кортежів відношення R з кортежами відношення S.

$$Q = R \times S = \{(r, s) \mid r \in R \wedge s \in S\}$$

Властивості: Комутативна, асоціативна.

4.3.4. Операція з'єднання (Join)

Означення: Нехай відношення R має схему $R(L, M)$, а відношення S — схему $S(N, P)$, і множини атрибутів M і N є **θ-порівнянними**. θ-з'єднанням відношень R і S за умовою $M\theta N$ (позначається $R[M\theta N]S$) називається відношення $Q(L, M, N, P)$, кортежі якого є з'єднаннями тих кортежів з R і S, для яких виконується умова $r[M]\theta s[N]$.

$$Q = R \bowtie_{M\theta N} S = \{(r, s) \mid r \in R \wedge s \in S \wedge r[M]\theta s[N]\}$$

Зауваження: Атрибути, за якими виконується з'єднання, повторюються в результаті.

Часткові випадки:

- **Еквіз'єднання (Equijoin):** З'єднання за умовою рівності.
- **Природне з'єднання (Natural Join):** Позначається $R * S$. Це еквіз'єднання за однойменними атрибутами, причому один з дубльованих стовпців видаляється з результату.

- **Напівз'єднання (Semijoin):** Результатом є проекція з'єднання на атрибути одного з відношень.

Властивості: Комутативність, асоціативність.

4.3.5. Операція ділення (Division)

Означення: Нехай задано відношення $R(M,N)$ та $S(K,L)$, для яких проекції $R[N]$ та $S[K]$ є сумісними. Діленням відношення R на відношення S за наборами атрибутів N і K (позначається $R[N \div K]S$) називається відношення $Q(M)$, що складається з таких кортежів $t \in R[M]$, образи $I_R(t)$ яких містять усі кортежі проекції $S[K]$.

$$Q = R[N \div K]S = \{t \mid t \in R[M] \wedge I_R(t) \supseteq S[K]\}$$

Поняття образу: Образом реляційного відношення $R(M,N)$ за кортежем $t_1 \in R[M]$ називається множина кортежів $t_2 \in R[N]$, для яких зчеплення (t_1, t_2) належить відношенню R. Позначається $I_R(t_1)$.

Приклад:

Нехай R:

A	B	C
a_1	b_1	c_1
a_1	b_1	c_2
a_1	b_3	c_2
a_2	b_1	c_4

Нехай S:

C	D
c_1	d_1
c_1	d_2
c_2	d_3

$$S[C] = \{c_1, c_2\}$$

Тоді $R[C \div C]S$:

A	B
a_1	b_1

(Оскільки тільки для пари (a_1, b_1) образ $I_R((a_1, b_1)) = \{c_1, c_2\}$ містить усі елементи $S[C]$).

4.4. Приклади застосування операцій реляційної алгебри

Розглянемо схему бази даних університету:

ФАКУЛЬТЕТ(#F, Назва, Декан, Корпус, Фонд)

КАФЕДРА(#D, #F, Назва, #ЗАВІДУВАЧ, Корпус, Фонд)

ВИКЛАДАЧ(#T, #D, Прізвище, Посада, Тел)

ГРУПА(#G, #D, Курс, Номер, Кількість, #КУРАТОР)

ПРЕДМЕТ(#S, Назва)

АУДИТОРІЯ(#R, Номер, Корпус, Місткість)

ЛЕКЦІЯ(#T, #G, #S, #R, Тип, День, Тиждень)

Запит 1. Вивести список усіх викладачів разом з їхніми телефонами.

$$П_{\text{Прізвище, Тел}}(\text{ВИКЛАДАЧ})$$

Запит 2. Хто є деканом факультету інформатики?

$$П_{\text{Декан}}(\sigma_{\text{Назва}='\text{інформатики}'}(\text{ФАКУЛЬТЕТ}))$$

Запит 3. Вивести назви факультетів разом із назвами їхніх кафедр.

$$П_{\text{ФАКУЛЬТЕТ.Назва, КАФЕДРА.Назва}}(\text{ФАКУЛЬТЕТ} \bowtie_{\#F=\#F} \text{КАФЕДРА})$$

Запит 4. Вивести список усіх викладачів факультету інформатики разом з номерами їхніх телефонів.

$$П_{\text{ФАКУЛЬТЕТ.Назва, Прізвище, Тел}}(\sigma_{\text{ФАКУЛЬТЕТ.Назва}='\text{Інформатики}'}($$
$$(\text{ФАКУЛЬТЕТ} \bowtie_{\#F=\#F} \text{КАФЕДРА}) \bowtie_{\#D=\#D} \text{ВИКЛАДАЧ}))$$

Запит 5. Вивести номери груп першого курсу кафедри ПМ разом із прізвищами кураторів.

$$П_{\text{Номер, Прізвище}}(\sigma_{\text{Назва}='\text{ПМ}' \wedge \text{Курс}=1}((\text{ГРУПА} \bowtie_{\#D=\#D} \text{КАФЕДРА})$$

$$\bowtie_{\#\text{КУРАТОР}=\#T} \text{ ВИКЛАДАЧ}))$$

Запит 6. Вивести список лекцій, на яких кількість студентів у групі перевищує місткість аудиторії.

$$\Pi_{\text{АУДИТОРІЯ.Номер, ГРУПА.Номер, ПРЕДМЕТ.Назва, ЛЕКЦІЯ.Тиждень, ЛЕКЦІЯ.День}}$$
$$(\sigma_{\text{ГРУПА.Кількість}>\text{АУДИТОРІЯ.Місткість}}(((\text{ЛЕКЦІЯ} \bowtie_{\#G=\#G} \text{ГРУПА}) \bowtie_{\#S=\#S} \text{ПРЕДМЕТ})$$
$$\bowtie_{\#R=\#R} \text{АУДИТОРІЯ}))$$

Запит 7. Хто з викладачів кафедри КСМ не читає лекцій? (Використання різниці)

$$\Pi_{\text{Прізвище}}(\sigma_{\text{Назва}='\text{КСМ}'}(\text{КАФЕДРА} \bowtie_{\#D=\#D} \text{ВИКЛАДАЧ}))$$
$$- \Pi_{\text{Прізвище}}((\sigma_{\text{Назва}='\text{КСМ}'}(\text{КАФЕДРА} \bowtie_{\#D=\#D} \text{ВИКЛАДАЧ})) \bowtie_{\#T=\#T} \text{ЛЕКЦІЯ})$$

Запит 8. Вивести номери викладачів, які викладають в усіх групах. (Використання ділення)

$$\Pi_{\#T}(\Pi_{\#T,\#G}(\text{ЛЕКЦІЯ}) \div \Pi_{\#G}(\text{ГРУПА}))$$

Запит 9. Вивести прізвища викладачів, які викладають принаймні в усіх групах першого курсу кафедри АСУ.

$$\Pi_{\text{Прізвище}}((\Pi_{\#T,\#G}(\text{ЛЕКЦІЯ})$$
$$\div \Pi_{\#G}(\sigma_{\text{Курс}=1 \wedge \text{Назва}='\text{АСУ}'}(\text{ГРУПА} \bowtie_{\#D=\#D} \text{КАФЕДРА}))) \bowtie_{\#T=\#T} \text{ВИКЛАДАЧ})$$

4.5. Властивості операцій реляційної алгебри

Ці властивості дозволяють оптимізувати запити.

1. **Комутативність, асоціативність та дистрибутивність** теоретико-множинних операцій (об'єднання, перетину, різниці).
2. **Ідемпотентність проекцій:** Якщо $L \subseteq M$, то

$$\pi_L(\pi_M(R)) = \pi_L(R)$$

3. **Дистрибутивність проекції:**
   a) $\pi_N(R \cup S) = \pi_N(R) \cup \pi_N(S)$
   b) $\pi_N(R \cap S) = \pi_N(R) \cap \pi_N(S)$
   c) $\pi_N(R - S) = \pi_N(R) - \pi_N(S)$

d) $\pi_N(R \times S) = \pi_N(R) \times \pi_N(S)$

e) $\pi_K(\sigma_F(R)) = \sigma_F(\pi_K(R))$, якщо в умові F використовуються лише атрибути з K.

4. **Ідемпотентність (комутативність) селекцій:**

$$\sigma_F(\sigma_G(R)) = \sigma_G(\sigma_F(R)) = \sigma_{F \wedge G}(R)$$

5. **Комутативність селекції з декартовим добутком:**

a) $\sigma_F(R \times S) = \sigma_F(R) \times S$, якщо F стосується лише атрибутів R.

b) $\sigma_{F1 \wedge F2}(R \times S) = \sigma_{F1}(R) \times \sigma_{F2}(S)$, якщо $F1$ стосується лише R, а $F2$ – лише S.

6. **Дистрибутивність селекції з теоретико-множинними операціями:**

a) $\sigma_F(R \cup S) = \sigma_F(R) \cup \sigma_F(S)$

b) $\sigma_F(R \cap S) = \sigma_F(R) \cap \sigma_F(S)$

c) $\sigma_F(R - S) = \sigma_F(R) - \sigma_F(S)$

(Для (б) і (в) атрибути умови F повинні входити в обидва відношення).

7. **Комутативність селекції і з'єднання:**

a) $\sigma_G(R \bowtie_F S) = \sigma_G(R) \bowtie_F S$, якщо G стосується лише атрибутів R.

b) $\sigma_{G1 \wedge G2}(R \bowtie_F S) = (\sigma_{G1}(R)) \bowtie_F (\sigma_{G2}(S))$, якщо $G1$ стосується R, а $G2$ – S.

8. **Комутативність та асоціативність** декартового добутку та з'єднання.

9. **Дистрибутивність з'єднання з теоретико-множинними операціями:**

a) $R \bowtie (S \cup T) = (R \bowtie S) \cup (R \bowtie T)$

b) $R \bowtie (S \cap T) = (R \bowtie S) \cap (R \bowtie T)$

c) $R \bowtie (S - T) = (R \bowtie S) - (R \bowtie T)$

Реляційна алгебра є потужним формальним апаратом для роботи з даними. Її операції дозволяють виконувати складні маніпуляції з реляційними відношеннями, формуючи основу для мови запитів SQL. Розуміння властивостей цих операцій є ключовим для оптимізації виконання запитів та побудови ефективних баз даних. Наведений матеріал охоплює

фундаментальні аспекти реляційної алгебри та служить надійною основою для подальшого вивчення теорії та практики баз даних.

§5. Мова структурованих запитів

5.1. Історія виникнення мови структурованих запитів.

Історія мови структурованих запитів (SQL) нерозривно пов'язана з розвитком реляційних баз даних. Це був революційний крок у способах зберігання, організації та маніпулювання величезними обсягами інформації. До появи реляційних баз даних і SQL, системи управління базами даних (СУБД) були переважно **ієрархічними** або **мережевими**. Ці моделі були досить складними у використанні, їх було важко змінювати (модифікувати схему даних), а запити до них вимагали написання складного процедурного коду. Кожна зміна у структурі даних могла призвести до переписування значної частини програмного забезпечення, що працювало з цією базою.

Саме в цей період зростала потреба у більш гнучкому, простому та математично обґрунтованому способі роботи з даними.

Ключова подія, яка стала основою для SQL, відбулася у **1970 році**, коли **Едгар Ф. Кодд (Edgar F. Codd)**, математик і дослідник, який працював у компанії **IBM**, опублікував свою знакову статтю: "A Relational Model of Data for Large Shared Data Banks" (Реляційна модель даних для великих спільних банків даних).

Основні ідеї Кодда полягали у наступному.

- **Реляційна модель:** Кодд запропонував представляти дані у вигляді **таблиць** (які він називав "відношеннями" або "реляціями"), що складаються з рядків (кортежів) і стовпців (атрибутів). Ця проста, але геніальна ідея значно спрощувала логічне сприйняття даних.

- **Математична основа:** Він наголошував на тому, що реляційна модель має бути заснована на **математичній теорії множин і реляційній алгебрі/численні**. Це забезпечувало чітку, несуперечливу логіку для виконання операцій над даними.
- **Декларативний підхід:** Кодд передбачав необхідність створення **декларативної мови запитів**, де користувач описує, *що* він хоче отримати (результат), а не *як* його отримати (процедурні кроки). Це відрізняло його підхід від існуючих процедурних мов.

Хоча Кодд не винайшов SQL безпосередньо, його реляційна модель стала теоретичним фундаментом, на якому базується SQL.

Після публікації Кодда, дослідницька група в IBM San Jose Research Laboratory, до якої входили Дональд Чемберлін (Donald D. Chamberlin) та Реймонд Бойс (Raymond F. Boyce), взялася за створення мови запитів для реляційної моделі.

У **1973** році вони розпочали розробку мови під назвою **SEQUEL** (Structured English Query Language – Структурована англійська мова запитів). Назва була натвірнена ідеєю "секвенціального" (послідовного) доступу до даних.

Наступного року була опублікована стаття "SEQUEL: A Structured English Query Language", яка детально описувала синтаксис і можливості нової мови.

Ця мова, хоч і мала назву SEQUEL, заклала більшість фундаментальних синтаксичних елементів, які ми знаємо як SQL сьогодні: SELECT, FROM, WHERE, INSERT, UPDATE, DELETE.

Проект **System R** (1974-1979) в IBM був першою масштабною реалізацією реляційної бази даних, яка використовувала мову SEQUEL.

Приблизно у 1978 році, через існуючу торгову марку "SEQUEL" у авіаційної компанії Hawker Siddeley, IBM була змушена змінити назву своєї мови. Так **SEQUEL** став **SQL** (вимовляється як "сіквел" або "ес-к'ю-ел").

System R довів життєздатність реляційної моделі та ефективність декларативної мови запитів у реальному середовищі. Це був критичний крок від теоретичної концепції до практичної реалізації.

Хоча IBM першою розробила SQL, вона не стала першою компанією, яка вивела комерційний реляційний продукт на ринок.

У 1977 році компанія **Relational Software Inc.** (згодом перейменована на **Oracle Corporation**), заснована Ларрі Еллісоном, Бобом Майнером і Едом Оутсом, почала розробляти власну реляційну СУБД, натхненну публікаціями IBM про System R та SQL.

Вже у 1979 році Oracle випускає першу комерційно доступну реляційну СУБД для мінікомп'ютерів, що працювала на VAX-11/780. Цей продукт швидко здобув популярність завдяки своїй функціональності та відносній простоті використання.

У 80-ті роки інші великі гравці, такі як **IBM** (з DB2), **Sybase**, **Microsoft** (з SQL Server) та інші, також виходять на ринок зі своїми реалізаціями SQL, конкуруючи за частку ринку.

Швидке поширення різних реалізацій SQL призвело до проблеми несумісності. Кожен виробник додавав свої розширення, що ускладнювало перенесення програмного забезпечення між різними СУБД. Це спричинило потребу у стандартизації.

- **1986 рік: Американський національний інститут стандартів (ANSI)** офіційно публікує перший стандарт **SQL**. Це був важливий крок до уніфікації мови.
- **1987 рік: Міжнародна організація зі стандартизації (ISO)** приймає цей стандарт.
- **Подальші стандарти:** З того часу SQL постійно розвивався, і ISO/IEC публікувала оновлені стандарти, додаючи нові можливості та вдосконалюючи синтаксис. Найбільш відомі версії:
    - **SQL-89**
    - **SQL-92 (або SQL2)**: Дуже значний стандарт, який ввів багато нових функцій.
    - **SQL:1999 (SQL3)**: Включив об'єктно-реляційні можливості.

- **SQL:2003, SQL:2008, SQL:2011, SQL:2016, SQL:2023**: Постійні оновлення, що додають підтримку XML, аналітичних функцій, віконних функцій, JSON, графових даних тощо.

Повна історія версій стандартів-ревізій SQL наведена у таблиці

Рік	Назва	Інша назва	Коментар
1986	SQL-86	SQL-87	Вперше оприлюднено ANSI. Ратифіковано ISO в 1987.
1989	SQL-89	FIPS 127-1	Незначні зміни.
1992	SQL-92	SQL2	Вагомі зміни.
1999	SQL:1999	SQL3	Додано регулярні вирази, рекурсивні запити, тригери та деякі об'єктно-орієнтовані нововведення.
2003	SQL:2003	SQL 2003	Впроваджені розширення для роботи з XML-даними.
2006	SQL:2006	SQL 2006	ISO/IEC 9075-14:2006. Функціональність роботи з XML-даними значно розширено. З'явилась можливість сумісного використання в SQL та XQuery.
2008	SQL:2008	SQL 2008	Вдосконалені можливості віконних функцій, усунуто деякі неоднозначності стандарту SQL:2003. Легалізовано ORDER BY поза визначенням курсору. Додано тригери INSTEAD OF. Додано заяви TRUNCATE.
2011	SQL:2011	SQL 2011	Додає часові дані. Покращення функцій вікон та пропозиції FETCH.

2016	SQL:2016	SQL 2016	Додає рядки підрівнювання посилань, поліморфні функції таблиці, JSON.

Сучасність та Майбутнє SQL

Сьогодні SQL залишається **де-факто стандартом** для роботи з реляційними базами даних. Він лежить в основі таких гігантів, як Oracle Database, Microsoft SQL Server, PostgreSQL, MySQL, IBM Db2, SQLite та багатьох інших.

Ключові фактори довголіття та успіху SQL:

- **Декларативність:** Легкість, з якою можна описати бажаний результат, не вдаючись у деталі реалізації.
- **Простота (відносна):** Основні команди SQL є досить інтуїтивними та схожими на англійську мову.
- **Потужність:** Здатність виконувати складні операції над великими обсягами даних.
- **Математична основа:** Реляційна модель забезпечує цілісність та консистентність даних.
- **Універсальність:** Застосовується в усіх галузях, де потрібно зберігати та аналізувати структуровані дані.

Незважаючи на появу NoSQL баз даних та інших парадигм, **SQL залишається незамінним інструментом** для роботи зі структурованими даними. Його принципи, закладені Е.Ф. Коддом та реалізовані у перших системах, досі є актуальними та продовжують розвиватися з новими викликами та технологіями. Це справді одна з найважливіших мов у світі інформаційних технологій.

5. 2. Архітектура SQL: класифікація елементів мови

5.2.1. Основні компоненти SQL

- **Пункти** (Clauses) - складові частини інструкцій (SELECT, FROM, WHERE)
- **Вирази** (Expressions) - генерують скалярні значення або таблиці
- **Предикати** (Predicates) - умови, що повертають TRUE/FALSE/UNKNOWN
- **Запити** (Queries) - операції отримання даних
- **Інструкції** (Statements) - команди, що виконують дії над даними

5.2.2. Синтаксичні особливості мови SQL

- Крапка з комою (;) як роздільник інструкцій – ознака завершення конструкції;
- Ігнорування пробілів та переносів рядків – одна конструкція може розташовуватися у декількох рядках;
- Регістронезалежність ключових слів (в більшості СУБД);
- Використання коментарів (/* */ та --);

5.3. Огляд підмов SQL: DDL, DML, DCL, TCL

Data Definition Language (DDL) – набір конструкцій, які відповідають за структуру бази даних:

- CREATE - створення об'єктів;
- ALTER - модифікація об'єктів;
- DROP - видалення об'єктів.

Data Manipulation Language (DML) - конструкції маніпулювання даними. Операції з даними:

- SELECT - вибірка даних;
- INSERT - додавання даних;
- UPDATE - оновлення даних;
- DELETE - видалення даних.

Data Control Language (DCL)

Управління правами доступу до даних:

- GRANT - надання привілеїв
- REVOKE - відкликання привілеїв

Transaction Control Language (TCL)

Керування транзакціями:

- BEGIN TRANSACTION - початок транзакції
- COMMIT - підтвердження змін
- ROLLBACK - скасування змін

5.4. Структура навчальної бази даних UNIVERSITY

Опишемо предметну область «заклад вищої освіти UNIVERSITY» наступними таблицями:

FACULTY (**IDF**, Name, Dean, Building, Fund)
DEPARTMENT (**IDD**, IDF, Name, Head, Building, Fund)
TEACHER (**IDT**, IDD, Name, Post, Tel)
GROUP (**IDG**, IDD, Course, Number, Quantity, IDCurator)
ROOM (**IDR**, Number, Building, Seats)
SUBJECT (**IDS**, Name)
LECTURE (IDT, IDG, IDS, IDR, Type, Day, Week)
Червоним кольором виділені зовнішні ключі таблиць, а чорним їх первинні ключі. Зв`язки між таблицями реалізуються наступним чином

- Факультет ↔ Кафедра (1:N)
- Кафедра ↔ Викладач (1:N)
- Кафедра ↔ Група (1:N)
- Викладач ↔ Лекція (1:N)
- Група ↔ Лекція (1:N)
- Аудиторія ↔ Лекція (1:N)

- Предмет ↔ Лекція (1:N)

5.4. Основи конструкції SELECT: вибірка даних

5.4.1. Базова структура запиту

SELECT стовпці

FROM таблиці

[WHERE умова];

Приклади базових запитів

1) Скласти список всіх факультетів

```
SELECT Name FROM FACULTY;
```

2) Відобразити всі дані з таблиці

```
SELECT * FROM TEACHER;
```

3) Вивести для кожного викладача його посаду і телефон

```
SELECT Name, Post, Tel FROM TEACHER;
```

- Фрази SELECT та FROM є обов'язковими означає вибірку всіх стовпців
- Порядок стовпців у результаті відповідає порядку у SELECT

Робота з унікальними значеннями та перейменування стовпців

У фразі SELECT можна визначати список імен стовпців. Передбачається, що результат виведення буде впорядкований за стовпцями відповідно до того, як розташовані імена у фразі. Наприклад,

SELECT ***DISTINCT*** Course, Number, Quantity

FROM GROUP;

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;
```

Для отримання унікальних комбінацій курсів та номерів груп, формуємо конструкцію

```
SELECT DISTINCT Course, Number FROM GROUP;
```

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, в наступному запиті перевизначаться імена обох стовпців:

```
SELECT Name AS Назва_факультету,
       Dean AS Декан_факультету
      FROM  FACULTY;
```

Варто зауважити, ключове слово DISTINCT корисне для аналітики та побудови звітів, де потрібні унікальні значення. Перейменування робить результати запитів більш зрозумілими для кінцевих користувачів.

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 ньому 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 університету:

```
SELECT Name FROM TEACHER
WHERE Post="Професор";
```

Умова - це вираз, який повертає значення *true* або *false*. Розрізняють прості і складені умови. Проста умова складається із виразу, оператора та значення, з яким порівнюють значення виразу. Складена умова має щонайменше дві прості умови, об`єднані логічними операціями OR чи AND.

Оператори - це конструкції, які використовуються для вказівки дій над операндами виразу.

Оператори порівняння, які використовуються у записі умов:

- = <> != - рівність/нерівність
- < <= > >= - порівняння
- BETWEEN - діапазон значень
- IN - належність множині
- LIKE - порівняння з шаблоном
- IS NULL - перевірка на NULL

Логічні оператори

- AND - логічне І
- OR - логічне АБО
- NOT - заперечення

Приклад. Отримати список викладачів, які працюють на посаді професора або асистента:

```
SELECT Name, Post
FROM TEACHER
WHERE  Post  IN (“професор”, “асистент”);
```

Знайти факультет, в назві яких є слово «інформатика» (пошук за шаблоном)

```
SELECT Name FROM FACULTY WHERE Name LIKE '%інформатики%';
```

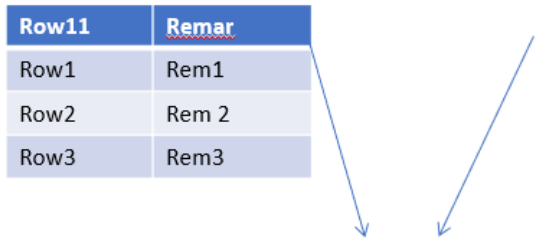
Оскільки ми працюємо із реляційною моделлю, то виникає необхідність використовувати декілька таблиць одночасно у одному запиті. Щоб вибрати дані з кількох таблиць, необхідно перелічити їхні імена у фразі FROM. Якщо у фразі WHERE не зазначено умову з'єднання таблиць, то обчислюється ***Декартів добуток*** усіх таблиць із фрази FROM. Наприклад, задано дві таблиці

Table 1 Table 2

Row11	Remar
Row1	Rem1
Row2	Rem 2
row3	Rem3

Row21	Remar
Row 1	Remar 1
Row 2	Remar 2
Row 3	Remar 3

Тоді їх декартів добуток – це комбінація записів першої таблиці із всіма записами другої таблиці



Row 11	Table1. Remar	Row21	Table 2.Remar
Row1	Rem1	Row 1	Remar 1
Row1	Rem1	Row 2	Remar 2
Row1	Rem1	Row 3	Remar 3
Row2	Rem 2	Row 1	Remar 1
Row2	Rem 2	Row 2	Remar 2
Row2	Rem 2	Row 3	Remar 3
Row3	Rem3	Row 1	Remar 1
Row3	Rem3	Row 2	Remar 2
Row3	Rem3	Row 3	Remar 3

Як бачимо, кожен рядок першої таблиці з'єднався з кожним рядком другої. За-звичай таблиці поєднуються за певної умови. Наприклад, для визначення назв факультетів та відповідних їм кафедр слід записати:

```
SELECT    FACULTY.Name, DEPARTMENT.Name
FROM      FACULTY, DEPARTMENT
WHERE     FACULTY.IDF = DEPARTMENT.IDF;
```

9.3. Використання псевдонімів

sql

```
-- Скорочені псевдоніми для таблиць
SELECT f.Name AS Факультет, d.Name AS Кафедра
FROM FACULTY f, DEPARTMENT d
WHERE f.IDF = d.IDF;
```

Якщо є стовпець, який має одне і те ж ім'я в об'єднувальних таблицях, то при посиланні на той або інший стовпець в запиті необхідно уточнювати їх іменами таблиць.

Мова SQL надає можливість пов'язувати з кожною таблицею деякий короткий псевдонім, і надалі посилатися на таблицю по ньому. Зіставлення таблиці і псевдоніму робиться за допомогою ключового слова FROM. Наприклад:

```
SELECT f.Name, d.Name
FROM Faculty f, DEPARTMENT d
WHERE f.IDF=d.IDF ;
```

Приклади.

1.Визначити всі кафедри факультету прикладної математики.

Мова SQL:

```
SELECT  d.Name
FROM  FACULTY f, DEPARTMENT d
WHERE  f.IDF=d.IDF AND f.Name=“ФПМ”;
```

У термінах реляційної алгебри цей запит виглядає таким чином

((ФАКУЛЬТЕТ [IDF=IDF] КАФЕДРА) ФАКУЛЬТЕТ.Назва=«ФПМ»])

[КАФЕДРА.Назва]

2.Визначити всіх викладачів кафедри МПУіК та їхні телефонні номери.

Мова SQL:

```
SELECT    t.Name, Tel
FROM  DEPARTMENT d, TEACHER t
WHERE    d.IDD = t. IDD  AND d.Name= “МПУіК”;
```

У термінах реляційної алгебри:

((КАФЕДРА[IDD=IDD]ВИКЛАДАЧ)[КАФЕДРА.Назва = “МПУіК"])[Прізвище, Тел]

3.Вивести список усіх викладачів ФПМ разом з їхніми теле-фонними номерами.

Мова SQL:

SELECT t.Name, Tel

FROM FACULTY f, DEPARTMENT d, Teacher t WHERE f.IDF=d.IDF AND d.IDD=t.IDD AND f.Name= “ФПМ”;

Реляційна алгебра:

(((ФАКУЛЬТЕТ[IDF = IDF] КАФЕДРА) [IDD =IDD] ВИКЛАДАЧ) [ФАКУЛЬТЕТ. Назва = “ФПМ"]) [Прізвище, Тел]

Функції в мові SQL дозволяють виконувати різні обчислювальні операції, які не можливо виразити стандартними засобами мови. Із безлічі наявних функцій ми розглянемо тільки, так звані, агрегатні функції, які включені в стандарт ANSI. До них відносяться такі:

- **COUNT** – кількість значень вказаного стовпця;
- **SUM** – сума всіх значень стовпця;
- **AVG** – середнє значення всіх значень стовпця;
- **MAX** – найбільше значення в стовпці;
- **MIN** – найменше значення в стовпці.

Приклади. Визначити кількість кафедр на факультеті прикладної математики.

SELECT COUNT(*)

FROM FACULTY f, DEPARTMENT d

WHERE f.IDF = d.IDF AND f.Name = «прикладної математики“;

Обчислити кількість предметів, за якими навчаються.

SELECT COUNT(*) AS Number_Of_Subjects

FROM SUBJECT;

Визначити найбільший фонд із фінансування серед кафедр факультету комп`ютерних наук.

SELECT MAX(d.Fund) AS MAX_Department_Fund_In_IT_Faculty

FROM FACULTY f, DEPARTMENT d

WHERE f.IDF = d.IDF AND f.Name = “КН“;

Можна задавати кілька агрегатних функцій, що використовуються у виразах.

Наприклад, визначити місткість усіх аудиторій корпусу 5, кількість місць у найменшій та найбільшій аудиторіях, середню місткість.

SELECT SUM(Seats) AS Total_Number_Of_Seats_In_Building_5,

MIN(Seats) AS Seats_In_The_Smalest_Room ,

MAX(Seats) AS Seats_In_The_Largest_Room,

SUM(Seats)/COUNT(*) AS Average_Number_Of_Seats

FROM ROOM

WHERE Building= 5;

Варто відмітити, якщо в запиті немає фрази GROUP BY, то використання імен стовпців разом із агрегатними функціями у фразі SELECT є неприпустимим. Наприклад, у результаті виконання запиту

SELECT Building, SUM(Seats) AS Total_Number_0f_Seats_In_Building

FROM ROOM;

Dynamic SQL Error -SQL error code = -104 -invalid column reference

Оскільки аргументом агрегатної функції має бути стовпець значень (одне поле чи вираз, аргументом в якому є поле), то їх не можна використовувати у фразі WHERE. В результаті виконання запиту

SELECT Name

FROM DEPARTMENT

WHERE Fund = MAX(Fund);

отримаємо повідомлення про помилку.

Є особливості використання агрегатної функції COUNT(). У стовпці-аргументі агрегатної функції перед застосуванням будь-якої функції, окрім C0UNT(*), виключаються всі невизначені значення. Сформулювавши запит

SELECT COUNT(Building), COUNT(DISTINCT Building), COUNT(*)
FROM ФАКУЛЬТЕТ;

можна отримати, наприклад, таку відповідь:

COUNT(Building) COUNT(DISTINCT Building) C0UNT(*)
15 11 17

Результат виконання запиту слід інтерпретувати так: усього є 17 факультетів; для 15 з них задані значення корпусу (два факультети в полі корпусу містять NULL), і серед цих 15 значень корпусів лише 11 є різними, тобто неповторюваними.

Записи у таблицях реляційної бази даних ніяким чином не впорядковані, не систематизовані. Однак, на вимогу користувача це можна зробити, використовуючи відповідні розділи конструкції SELECT.

Розділ **GROUP BY** групує рядки з однаковими значеннями вказаних стовпців, дозволяючи застосовувати агрегатні функції до кожної групи окремо.

```
SELECT стовпці_групування, агрегатні_функції
FROM таблиці
WHERE умови
GROUP BY стовпці_групування;
```

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. Якщо в ключовому слові SELECT, присутні імена стовпців, то вони мають бути тими стовпцями, за допомогою яких проводиться групування.

За наявності фрази GROUP BY кожний вираз у списку фрази SELECT повинен набувати єдиного значення для всієї групи, тобто він може бути:

- ♦ константою;
- ♦ агрегатною функцією;

- ♦ таким самим виразом, що й у фразі GROUP BY;
- ♦ виразом, що побудований з перелічених вище виразів.

Приклади. Підрахувати кількість студентів на кожній кафедрі ФПМ

```
SELECT d.Name, SUM(g.Quantity) AS Student_Count
FROM FACULTY f, DEPARTMENT d, GROUP g
WHERE f.IDF = d.IDF AND d.IDD = g.IDD
  AND f.Name = 'ФПМ'
GROUP BY d.Name;
```

Підрахувати кількість предметів кожного викладача

```
SELECT t.Name, COUNT(DISTINCT l.IDS) AS Subjects_Count
FROM TEACHER t, LECTURE l
WHERE t.IDT = l.IDT
GROUP BY t.Name;
```

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. 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.

Наприклад, необхідно вивести ті факультети, у яких фонд факультету перевищує сумарний фонд всіх кафедр:

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

Ключове слово **ORDER BY** дозволяє вказати імена стовпців, за якими необхідно відсортувати вихідні рядки, а також порядок їх сортування: за збільшенням - **ASC** або за зменшенням - **DESC**.

Вивести в порядку зростання кількості викладачів на всіх кафедрах навчального закладу.

```
SELECT     d.Name, COUNT(*)  AS Number_Of_Faculty_Teachers
FROM DEPARTMENT d, TEACHER t
WHERE  d.IDD = t.IDD
GROUP BY d.Name
ORDER BY  Number_Of_Faculty_Teachers;
```

Приклади. Скласти список кафедр, впорядкувавши його за кількістю викладачів на кожній кафедрі.

```
SELECT d.Name, COUNT(*) AS Teacher_Count
FROM DEPARTMENT d, TEACHER t
WHERE d.IDD = t.IDD
GROUP BY d.Name
ORDER BY Teacher_Count ASC;
```

Класти список викладачів, застосувавши впорядкування у проти алфавітному порядку.

```
SELECT Name FROM TEACHER ORDER BY Name DESC;
BY Course ASC, Number ASC;
```

Отже, конструкція відбору інформації за вказаними критеріями має структуру

SELECT ***список полів***
FROM **список таблиць**
WHERE **умови відбору записів**
GROUP BY **список полів**
HAVING **умови відбору груп**
ORDER BY ***список полів1*****;**

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:

1. Обчислюється декартовий добуток рядків з таблиць, вказаних у ключовому слові **from**;
2. До отриманої єдиної таблиці застосовуються умови з ключового слова **where**; ці умови формулюються таким чином, що їхня істинність перевірялася на рядку таблиці;
3. Отримані рядки групуються згідно умові у ключовому слові **group by**;
4. До згрупованих рядків застосовуються умови, задані у ключовому слові **having**; вибираються лише ті групи, які задовольняють ці умови.
5. Рядки (групи) упорядковуються згідно ключового слова **order by**;
6. Виводяться ті стовпці або вирази, які присутні у ключовому слові **select**.

На практиці застосування конструкцій SQL зазвичай комплексне і передбачає вкладеність – *підзапити*.

Розрізняють два типи підзапитів: простий та корельований. *Простий підзапит* - це підзапит, обчислення якого здійснюється незалежно від обчислення зовнішнього запиту. Такі підзапити обробляються «знизу вверх»: першим обробляється вкладений підзапит, а множина значень, отримана в результаті його виконання, використовується під час обробки зовнішнього запиту.

Корельований підзапит— це підзапит, обчислення якого залежить від процесу обчислення в зовнішньому запиті. Такі підзапити обробляються «зверху вниз»: спочатку вибирається поточний рядок із таблиці зовнішнього запиту й за значеннями його полів виконується обчислення підзапиту (тобто під час перевірки умов обчислення підзапиту використовуються значення полів із зовнішнього запиту). Далі перевіряється умова WHERE щодо входження поточного рядка зовнішнього запиту до результату.

Наведемо приклади. Кафедри в тому ж корпусі, що й кафедра ММ

```
SELECT Name
FROM DEPARTMENT
WHERE Building = (
   SELECT Building
   FROM DEPARTMENT
   WHERE Name = 'MM'
);
```

Знайти факультети з фондом, меншим за суму кафедр

```
SELECT Name
FROM FACULTY f
WHERE f.Fund < (
   SELECT SUM(Fund)
   FROM DEPARTMENT
   WHERE IDF = f.IDF
);
```

Вказати викладачів, які читають лекції

```
SELECT Name
FROM TEACHER t
WHERE EXISTS (
   SELECT *
   FROM LECTURE l
   WHERE l.IDT = t.IDT
);
```

Для формування умов різної складності до результатів виконання підзапиту можна застосовувати такі предикати, як ANY та ALL розміщують після символів однієї з операцій порівняння =, *, >, <, <=, >=, щоб перевірити, чи є предикат істинним принаймні для одного (для всіх) значень множини, заданої в дужках після слова ANY (ALL), стосовно елементу, записаного зліва від символів порівняння. Розглянемо приклад.

Приклади. Визначити кафедри, фонди яких більші, ніж хоча б у однієї з кафедр факультету математики та інформатики.

```
SELECT     Name
FROM  DEPATMENT
WHERE       Fund >ANY (SELECT    d.Fund
FROM  FACULTY f,  DEPARTMENT d
WHERE      f.IDF = d.IDF AND
f.Name = “математики та інформатики");
```

EXISTS є одноаргументним предикатом, що повертає значення TRUE, коли підзапит, до якого він застосовується, містить хоча б один рядок.

Приклад. Визначити викладачів, які читають хоча б один курс лекцій.

```
SELECT  Name
FROM TEACHER
WHERE      EXISTS (SELECT     *
FROM  LECTURE l
WHERE      l .IDT = TEACHER.IDT);
```

Результати виконання запитів можна використовувати аргументи теоретико-множинних операцій(UNION, INTERSECT, EXCEPT). Такі операції потрібно застосовувати до сумісних множин. Приклади застосування таких операцій наводимо нижче.

Об'єднання (UNION)

Вказати усі назви факультетів та кафедр

```
SELECT Name FROM FACULTY
UNION
SELECT Name FROM DEPARTMENT;
```

Перетин (INTERSECT)

Викладачі, які читають лекції в понеділок та вівторок

```
SELECT t.Name
FROM TEACHER t, LECTURE l
WHERE t.IDT = l.IDT AND l.Day = 'понеділок'
```

```
INTERSECT
SELECT t.Name
FROM TEACHER t, LECTURE l
WHERE t.IDT = l.IDT AND l.Day = 'вівторок';
```

Різниця (EXCEPT)

Викладачі, які читають «Бази даних», але не «Програмування»

```
SELECT t.IDT
FROM TEACHER t, LECTURE l, SUBJECT s
WHERE t.IDT = l.IDT AND l.IDS = s.IDS
  AND s.Name = 'Бази даних'
EXCEPT
SELECT t.IDT
FROM TEACHER t, LECTURE l, SUBJECT s
WHERE t.IDT = l.IDT AND l.IDS = s.IDS
  AND s.Name = 'Програмування';
```

Окрім конструкції SELECT до розділу DML мови SQL відноситься низка конструкцій, призначених для виконання модифікацій даних у таблицях. Окрім пошуку даних у таблицях можна вводити нові дані, змінювати введені дані, знищувати не потрібні дані.

Для додавання даних використовують конструкцію INSERT. Її синтаксис такий

INSERT INTO <ім'я таблиці> (<поле 1> [, <поле 2>]...)

VALUES (<значення 1> [, <значення 2>]...);

Важливо дотримуватись відповідності по послідовності, типу даних для кожного поля і його значення.

Приклад. Додавання нового факультету

```
INSERT INTO FACULTY (IDF, Name, Dean, Building, Fund)
VALUES (15, 'інформатики', 'Сидоров', 5, 25000);
```

Якщо є потреба скопіювати дані із однієї таблиці в іншу, то це можна зробити конструкцією

INSERT INTO <ім'я таблиці> (<список полів>)

SELECT <список полів>

FROM <ім'я таблиці>

WHERE <умова пошуку>;

Списки полів мають після ключового слова INSERT INTO мають співпадати із списком після ключового слова SELECT не тільки по кількості, а й по типу та змісту полів. Наприклад, скопіювати дані з інших таблиць

```
INSERT INTO Temporary (Name_Faculty, Name_Department, Teacher_Name)
SELECT f.Name, d.Name, t.Name
FROM FACULTY f, DEPARTMENT d, TEACHER t
WHERE f.IDF = d.IDF AND d.IDD = t.IDD;
```

Оновлення даних виконує конструкція UPDATE

UPDATE <ім'я таблиці>

SET <поле 1> = <вираз 1>

[, <поле 2> = <вираз 2>]...

[**WHERE** <умова пошуку>];

У якості <виразів> можна використати константи або вирази, значення яких є того ж типу, що і поле, значення якого змінюємо. Якщо ключове слово **WHERE** відсутнє, то змінюють значення усім записам таблиці. Якщо його використовують, то значення вказаних полів змінюються у тих записах, де <умова пошуку> приймає істинне значення.

Приклад. Оновлення фонду факультету (нове константне значення)

```
UPDATE FACULTY
SET Fund = 250300
WHERE Name = 'інформатики';
```

Приклад. Збільшення фонду на 10% (використання обчислювального виразу)

```
UPDATE FACULTY
SET Fund = Fund * 1.1;
```

Приклад. Оновлення на основі підзапиту (із використанням умови)

```
UPDATE DEPARTMENT
SET Fund = Fund * 1.05
WHERE IDF IN (
   SELECT IDF
   FROM FACULTY
   WHERE Name = 'інформатики'
);
```

Видалення даних виконується конструкцією DELETE

DELETE FROM <ім'я таблиці> [**WHERE** умова];

Як і в попередній конструкції ключове слово **WHERE** регулює видалення усіх записів таблиці (при його відсутності) чи групи записів, для яких вказана умова істинна. Приклад. Видалити предмети, для яких не читаються лекції

```
DELETE FROM SUBJECT
WHERE IDS NOT IN (SELECT DISTINCT IDS FROM LECTURE);
```

Приклад. Видалити усі записи таблиці

```
DELETE FROM SUBJECT;
```

Приклад. Видалення за умовою

```
DELETE FROM LECTURE
WHERE Day IN ('субота', 'неділя');
```

Конструкції DDL (Data Definition Language)

Конструкції цього розділу мови SQL працюють із об`єктами бази даних. З їх допомогою можна створити таблицю, представлення, індекс, тригер тощо, змінити структуру чи параметри створеного об`єкта та знищити його разом із даними, що в ньому є.

Конструкції створення розпочинаються із ключового слова CREATE, конструкції зміни структури чи параметрів – із ключового слова ALTER, а конструкції знищення об’єктів – DROP.

Створення бази даних

CREATE DATABASE <ім`я бази даних>;

CREATE DATABASE UNIVERSITY;

Створення таблиць має складнішу структури. Тут крім імені нової таблиці ретельно прописують її структуру, вказуючи назву кожного поля, його тип і обмеження на значення. Тут можуть бути присутні обмеження на комбінацію полів, визначення статусу полів як первинних чи зовнішніх ключів тощо.

Синтаксис

CREATE TABLE <ім'я таблиці> (<поле 1> <тип даних 1> [NOT NULL]
[, <поле 2> <тип даних 2> [NOT NULL]]...);

Приклад створення таблиці faculty

```
CREATE TABLE FACULTY (
   IDF INT PRIMARY KEY,
   Name VARCHAR(100) NOT NULL,
   Dean VARCHAR(100),
   Building VARCHAR(10),
   Fund DECIMAL(12,2)
);
```

Для зміни структури таблиць користуються конструкцією

ALTER TABLE <ім'я таблиці>
[ADD <поле> <означення поля>|
ALTER <поле> <параметри>|
DROP <поле> <параметри>];

У одній конструкції вказується одна дія, яка має бути виконана: добавлення поля із визначеними параметрами (**ADD** <поле> <означення поля>|), зміна параметрів існуючого поля (**ALTER** <поле> <параметри>) чи знищення поля (**DROP** <поле> <параметри>];)

Приклад. Додавання поля

ALTER TABLE FACULTY ADD COLUMN Phone VARCHAR(15);

Приклад. Зміна типу поля

ALTER TABLE FACULTY ALTER COLUMN Fund TYPE DECIMAL(15,2);

Приклад. Видалення поля

ALTER TABLE FACULTY DROP COLUMN Phone;

Видалення об`єктів виконується командами

DROP DATABASE <ім'я бази даних>; (видалення бази даних)

DROP TABLE <ім'я таблиці>; (видалення таблиці)

-Приклад видалення бази даних

DROP DATABASE UNIVERSITY;

Приклад. Видалення таблиці

DROP TABLE Faculty_Backup;

SQL залишається фундаментальною технологією для роботи з даними. Володіння як базовими, так і просунутими концепціями SQL дозволяє створювати ефективні, безпечні та масштабовані системи управління даними. Постійне вдосконалення стандартів та поява нових можливостей роблять SQL незамінним інструментом у арсеналі сучасного розробника.

Частина 2. Лабораторний практикум.

Лабораторна робота №1

Розробка структури бази даних. Застосування операцій реляційної алгебри

Завдання 1.

1. Формально описати предметну область (перелік тем містить Додаток1).
2. Виділити основні поняття предметної області та вказати взаємозв'язки між ними.
3. Вказати уточнення для понять предметної області.
4. Вказати можливі результати даної схеми.
5. Побудувати схему третьої нормальної форми.

Завдання 2. Використовуючи побудовану схема бази даних проілюструвати застосування операцій реляційної алгебри для пошуку та відбору інформації. Зокрема, вказати застосування операцій:

− проекція;
− обмеження (селекція);
− з`єднання;
− довільна комбінація операцій проекції, обмеження та з`єднання;
− об`єднання;
− перетин;
− різниця,
− ділення.

МЕТОДИЧНІ ВКАЗІВКИ

Першим етапом проектування бази даних будь-якого типу є аналіз предметної області, що закінчується побудовою інформаційної структури (концептуальної схеми). На даному етапі аналізуються запити користувачів, вибираються інформаційні об'єкти та їх характеристики і на основі проведеного аналізу формується структура предметної області, яка не залежить від програмного та технічного середовища, в якому буде реалізовуватися база даних. Аналіз предметної області доцільно розбити на три фази:

- аналіз концептуальних вимог та інформаційних потреб;
- виявлення інформаційних об'єктів та зв'язків між ними;
- побудова концептуальної моделі предметної області та проектування концептуальної схеми бази даних.

На етапі аналізу концептуальних вимог та інформаційних потреб необхідно вирішити наступні задачі:

- аналіз вимог користувача до бази даних (концептуальних вимог);
- виявлення задач, що мають місце, при обробці інформації, яка повинна бути представлена у базі даних (аналіз додатків);
- виявлення перспективних задач (перспективних додатків);
- документування результатів аналізу.

Вимогами користувачів до розроблюваної бази даних є, в загальному випадку, список запитів з вказанням їх інтенсивності та об'ємів даних. Ці вказівки опрацьовуються в діалозі з майбутнім користувачем бази даних. Тут же з'ясовуються вимоги до вводу, відновлення та корегування інформації. Вимоги користувачів уточнюються та доповнюються при аналізі перспективних додатків, що мають місце.

1. Формально описати предметну область.

Предметна область – **Автовокзал.**

У місті є автовокзал. Автовокзал обслуговує різні напрямки. На кожному напрямку є декілька маршрутів. Кожний маршрут обслуговує один перевізник. Перевізник забезпечує транспортом (автобусом) та водіями. По одному маршруту виконуються декілька рейсів. За їх дотриманням слідкує диспетчер автовокзалу. За кожним автобусом закріплено не більше 2 водіїв. Водії є працівниками перевізника. Вони отримують фіксовану заробітну плату, регулярно проходять медичний огляд, а транспорт проходить технічне обслуговування щомісяця. Кожний транспортний засіб та водій застраховані.

Автобуси курсують відповідно до розкладу. Для проїзду в автобусі потрібно придбати квиток у касі автовокзалу. Кожен автовокзал має свої каси, де позмінно працюють касири, отримуючи фіксовану заробітну плату. Перевірку придбаних квитків здійснюють контролери (вони теж працюють на автовокзалі і отримують фіксовану заробітну плату) при посадці на автобус. У квитку повинно вказуватися: звідки, куди прямує маршрут, № рейсу, тип транспорту, № місця та вартість проїзду. Ведеться облік проданих квитків.

Другим етапом аналізу предметної області складається з вибору інформаційних об'єктів, завдання необхідних властивостей для кожного об'єкта, виявлення зв'язків між об'єтами, виявлення обмежень, що накладаються на інформаційні об'єкти, типи зв'язків між ними, характеристики інформаційних об'єктів.

При виборі інформаційних об'єктів бажано намагатися відповісти на такі питання:

1. На які класи можна розбити дані, що підлягають зберіганню у базі даних?

2. Яке ім'я можна присвоїти кожному класу даних?

3. Які найбільш цікаві характеристики (з точки зору користувача) кожного класу даних можно виділити?

4. Які імена можна присвоїти вибраним наборам характеристик?

Виділення інформаційних об'єктів - процес ітеративний. Він здійснюється на основі аналізу інформаційних потоків та інтерв'ювання

споживачів. Характеристики інформаційних об'єктів визначаються тими ж методами.

2. Виділити основні поняття предметної області та вказати взаємозв`язки між ними.

Маршрути, перевізник, транспорт, працівники, квиток, страхування.

3. Вказати уточнення для понять предметної області.

Працівники: ПІБ працівника, де працює, рік народження, місце проживання, сімейний стан, освіта, посада, оклад, страховий поліс, номер та серія паспорта, кількість дітей.

Маршрут: № маршруту, початкова точка маршруту, кінцева точка маршруту, перевізник, що обслуговує маршрут, схема маршруту, вартість проїзду.

Рейс: № рейсу, маршрут, час початку, час завершення, який транспорт використовується, хто виконує рейс, ~~статус рейсу[скасовано, виконано, очікується]~~.

Перевізник: назва, адреса, телефон, директор, гол. бухгалтер, банк. реквізити.

Транспорт: VIN-номер, модель, марка, тип транспорту, ресурс двигуна, рік випуску, к-сть посадочних місць, кому належить транспорт, страховий поліс.

Страхування: назва компанії, адреса, телефон, контактна особа.

Квиток: номер, № рейсу, № місця, наявність багажу, дата, тип транспорту.

Розклад: № рейсу, дата виконання, статус рейсу [скасовано, виконано, очікується]

Уточнення:

Для працівників:

Медичний огляд: хто пройшов, коли пройшов, результати, рекомендації.

Технічний огляд: який ТЗ проходив, коли проходив, результати, рекомендації.

Страхування працівників: хто застрахований, де застрахований, які умови страхування, термін дії договору.

Страхування ТЗ: що застраховано, ким застраховано, де застрахований, які умови страхування, термін дії договору.

1. Які маршрути виконуються перевізником «Денисівка»? (Маршрут-Перевіник)

5. Вказати можливі результати даної схеми.

2. Скласти список водіїв, які працюють у перевізника «Уратранс». (Перевізник-Працівник)

3. Скласти список працівників, у яких закінчується термін дії страхування. (Страхування працівників- Працівники)

4.Скільки рейсів виконується на маршруті Чернівці-Київ? (Рейс-Маршрут)

5. Список оформлених квитків за 12/09/2020.(Квиток)

6.Кількість працівників, яким необхідно пройти медичний огляд. (Медичний огляд)

7.Скласти список «вільних» ТЗ. (Транспорт-Рейс)

8. Скласти список маршрутів, вартість проїзду на яких не перевищую 250 гривень.(Маршрут)

9. Скласти список скасованих рейсів. (Розклад)

10. Порахувати кількість маршрутів для кожного перевізника.

11. Вказати маршрут, по якому здійснюється рейс найпізніше.

12. Скласти перелік транспортних засобів, випущених більше 10 років.

13. Скласти список водіїв, у яких в поточному місяці закінчується термін страхування/ медичного обстеження.

14. Порахувати загальну вартість проданих квитків на запропонований маршрут.

15. Скласти перелік маршрутів, які починаються у певній(вказаній) зупинці.

(Кольорами виділяємо важливі поняття, уточнення)

Заключна фаза аналізу предметної області складається з розробки її інформаційної структури (або концептуальної схеми).

В звичайних випадках для побудови концептуальної схеми використовуються традиційні методи агрегації та узагальнення. При агрегації декілька інформаційних об'єктів (елементів даних) об'єднується в один у відповідності з семантичними зв'язками між об'єктами. При побудові

інфологічних моделей можна використовувати мову *ER*-діаграм (від англ. Entity-Relationship, "сутність-зв'язок"). В них сутності зображуються позначеними прямокутниками, асоціації - позначеними ромбами або шестикутниками, атрибути - позначеними овалами, а зв'язки між ними - ненаправленими ребрами, над якими може проставлятися ступінь зв'язку (1 або буква, що заміняє слово "БАГАТО") і необхідне пояснення.

Розглянемо деякі риси моделі "сутність-зв'язок" (*ER*-моделі).

На використанні різновидів *ER*-моделі реалізується більшість сучасних підходів до проектування баз даних. Модель була запропонована Ченом (Chen) у 1976 р. Моделювання предметної області базується на використанні графічних діаграм, що включають невелике число різнорідних компонентів. У зв'язку з наочністю уявлення концептуальних схем баз даних *ER*-моделі знайшли широке застосування в системах CASE, що підтримують автоматизоване проектування реляційних баз даних.

Головними поняттями *ER*-моделі є сутність, зв'язок і атрибут. У діаграмах *ER*-моделі сутність подається у вигляді прямокутника, що містить ім'я сутності. При цьому ім'я сутності - це ім'я типу, а не деякого конкретного екземпляра цього типу. Для кращого розуміння ім'я сутності може супроводжуватися прикладами конкретних об'єктів цього типу.

Кожний екземпляр сутності повинен відрізнятися від будь-якого іншого екземпляра тієї ж сутності (ця вимога до повної міри аналогічна вимозі відсутності кортежів-дублікатів у реляційних таблицях).

Зв'язок - це асоціація, що графічно зображується та встановлюється між двома сутностями. Ця асоціація завжди є бінарною і може існувати між двома різними сутностями або між сутністю і нею ж самою (рекурсивний зв'язок). У будь-якому зв'язку виділяються два кінці (відповідно до існуючої пари сутностей, що зв'язуються), на кожному з яких вказується ім'я кінця зв'язку, ступінь кінця зв'язку (скільки екземплярів даної сутності зв'язується), обов'язковість зв'язку (тобто чи будь-який екземпляр даної сутності повинен брати участь у даному зв'язку). Зв'язок подається у вигляді лінії, що зв'язує дві

сутності або веде від сутності до неї ж самої. Про це в місці "стикування" зв'язку із сутністю вказує триточковий вхід у прямокутник сутності, якщо для цієї сутності в зв'язку можуть використовуватися багато (many) екземплярів сутності, і одноточковий вхід, якщо в зв'язку може брати участь тільки один екземпляр сутності. Обов'язковий кінець зв'язку зображується суцільною лінією, а необов'язковий - переривчастою лінією.

Атрибутом сутності є будь-яка деталь, що служить для уточнення, ідентифікації, класифікації, числової характеристики або вираження стану сутності. Імена атрибутів заносяться в прямокутник, що зображує сутність, під ім'ям сутності і зображуються малими буквами, можливо, із прикладами. Наприклад, унікальним ідентифікатором сутності є атрибут, комбінація атрибутів, комбінація зв'язків або комбінація зв'язків і атрибутів, що унікально відрізняє будь-який екземпляр сутності від інших екземплярів сутності того ж типу.

Таким чином, зв'язки можуть представлятися різними способами, з яких ми будем використовувати тільки два: діаграма *ER*-екземплярів і діаграма *ER*-типів. Діаграми *ER*-екземплярів відображають зв'язки між екземплярами сутностей. Діаграми *ER*-типів відображають зв'язки між типами сутностей.

Отже, першою задачею, яку необхідно розв'язати при розробці *ER*-моделі, є формування сутностей, що необхідні для описання предметної області. Іншими словами, необхідно вказати ті типи об'єктів (тобто набори подібних об'єктів), про які в системі повинна накопичуватися інформація. Це означає, що за підсумками аналізу предметної області потрібно мати повну або достатньо повну уяву про реалізовані в системі запити. Разом з тим необхідно врахувати, що при кваліфікованій експлуатації системи у більшості випадків у користувача виникає бажання розширити систему запитів. У зв'язку з цим, при розробці *ER*-схеми, з одного боку зручно твердо прив'язатися до обраної в результаті аналізу предметної області системи запитів, а з іншого боку, бажано розглядати задачу в більш широкому плані, з урахуванням перспектив подальшого нарощування можливостей системи. Для кожної сутності

необхідно обрати атрибут, що її однозначно ідентифікує або сукупність атрибутів, які будуть виступати ключем відповідного відношення. Необхідно врахувати, що ключ повинен не тільки виконувати задачу ідентифікації, але і по можливості, включати в свій склад мінімальне число атрибутів. У зв'язку з цим в процесі проектування вибрані в якості ключів атрибути можуть переглядатися.

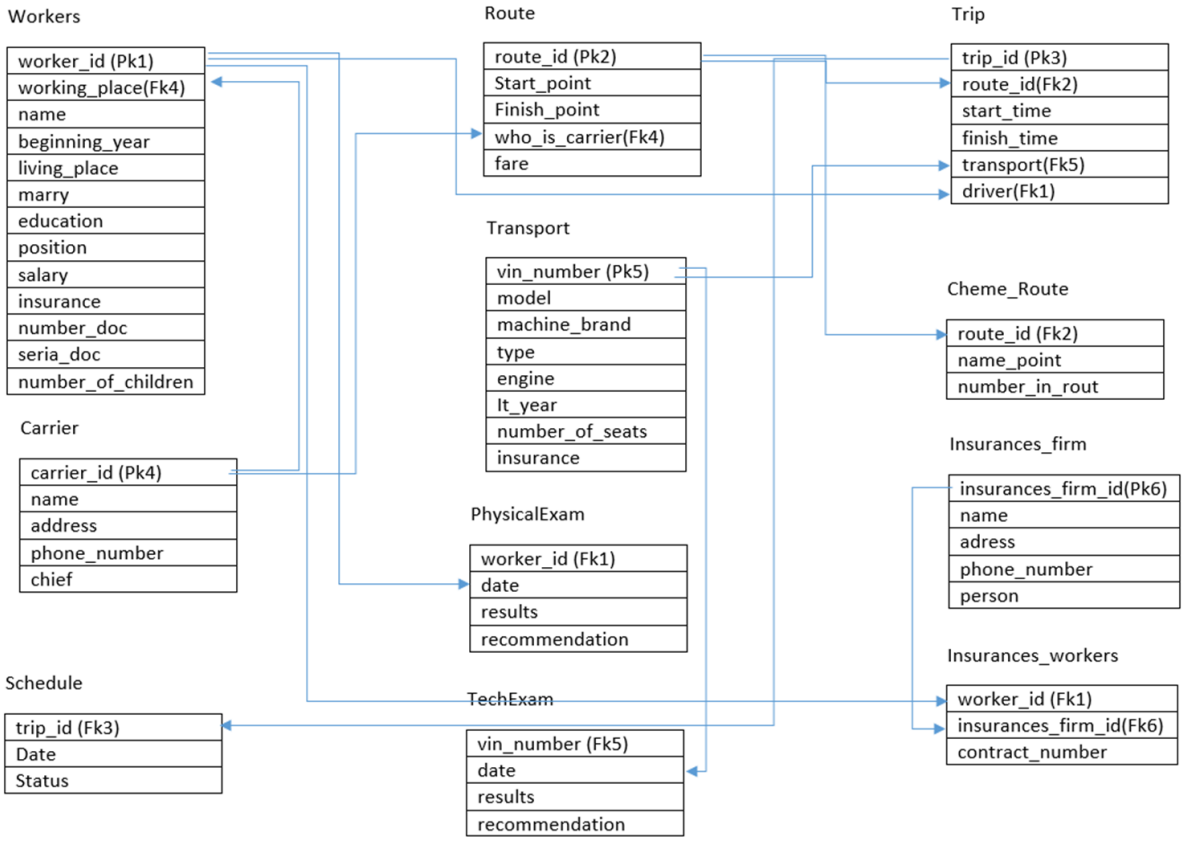


Схема БД.

Завдання 2. Застосування операцій реляційної алгебри у базі даних. Використовуючи побудовану схема бази даних (завдання 1) проілюструвати застосування операцій реляційної алгебри для пошуку та відбору інформації . Зокрема, вказати застосування операцій:

- **проекція;**
- **обмеження (селекція);**
- **з`єднання;**
- ***їх комбінування***;

- **об`єднання;**
- **перетин;**
- **різниця;**
- **ділення.**

Вхідні дані: Схема БД та перелік можливих результатів схеми (завдання1)

- **Проекція.**

Скласти список назв усіх перевізників

Carrier[name];

Скласти перелік усіх маршрутів (від-до)

Route[route_id,start_poin,finish_point];

- **З`єднання;**

Скласти перелік маршрутів кожного перевізника

(Carrier[carrier_id=who_is_carrier]Route)[name,route_id,start_point, finish_point];

Скласти перелік працівників кожного перевідника

(Carrier[carrier_id=working_place]Workers)[Workers.name, Carrier.name];

Скласти перелік усіх зупинок для усіх маршрутів

(Route[route_id=route_id]Scheme_Route)[Route.route_id, number_in_route,name];

Для кожного працівника вказати страхову фірму.

((Workers[worker_id=worker_id]Insurance_Workers)[insurance_firm_id= insurance_firm_id]Insurance_firm)[Workers.name,Insurance_firm.name];

- **Обмеження (селекція);**

Скласти список працівників, які почали працювати у 2010 році

Workers[beginning_year=2010][name];

Скласти перелік усіх водіїв

Workers[position=”водій”][name];

Скласти список ТЗ із об`ємом двигуна 2,0.

Transport[engine=2.0][model, machine_brand]

Скласти список скасованих рейсів

Schedule[status=”CANCELED”][trip_id]

- **Комбінації обмеження, селекції, проекції**

Скласти перелік тих перевізників, які мають транспорт, випущений до 2000 року

((Carrier[carrier_id=carrier]Transport)[it_year<2000])[name];

Визначити тих працівників, які проходили медичний огляд минулого року з рекомендацією повторного діагностування

((Worker[worker_id=worker_id]PhysicalExam)[recommendation=”повторне діагностування”&Year(date)=Year(Now())-1])[name];

Year(date), month(date),day(date)

Скласти список водіїв, які обслуговують маршрут№7 на Мерседесах.

(((Workers[worker_id=driver]Trip)[transport=VIN_number]Transport)[route_id=7, machine_brand=”Mersedes”])[Workers.name]

- **Перетин**

Скласти список марок машин, які є у перевізника «Комфорт» та «Вітерець».

(transport [t.carrier_id=c.carrier_id] carrier)[c.name=”Комфорт” macine_brend][macine_brand]

∩ transport [t.carrier_id=c.carrier_id] carrier)[c.name=”Вітерець” macine_brend][macine_brand];

- **Різниця;**

Скласти список тих транспортних засобів, які не проходили технічний огляд минулого місяця.

Transport[vin_number]- tech_exam[month(exam_data)=month(now())-1][vin_number];

Скласти список маршрутів, які виконуються не з Чернівців

- **Об`єднання**

Скласти перелік маршрутів перевізника «Денисівка» та «ПП Воловик»

(Route[who_is_carrier=carrier_id]carrier)[name=”Денисівка”][route_id]U
(Route[who_is_carrier=carrier_id]carrier)[name=”ПП Воловик”]
[route_id]

- **Ділення**

Скласти список тих страхових компаній, які страхували працівників усіх перевізників

(Insurance_worker[Insurance_firm_id,worker_id][**firm_id ÷ firm_id**]
Insurance_firm)[Insurance_firms.name];

Лабораторна робота №2

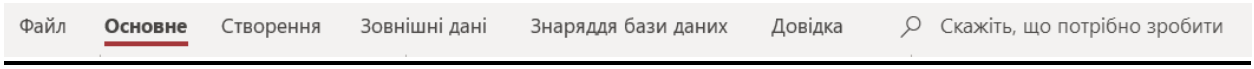
Реалізація схеми бази даних у MS Access.

ЗАВДАННЯ

1. Створити базу даних.
2. Створити структуру таблиці.
3. Ввести дані в таблицю.
4. Закрити таблицю і базу даних.
5. Відкрити базу даних і таблиці бази даних.
6. Змінити структуру таблиць бази даних.
7. Відредагувати, вилучити і добавити нові записи таблиці.
8. Переглянути таблиці.
9. Здійснити пошук даних.
10. Відсортувати записи.
11. Створити первинні ключі.
12. Відфільтрувати дані.

МЕТОДИЧНІ ВКАЗІВКИ

Після завантаження Microsoft ACCESS з'являється прикладне вікно ACCESS, яке є робочим простором, де розташовуються інші вікна програми. У верхній частині екрана розміщений рядок головного меню:



Крім нього також появиться діалогове вікно створення або відкриття існуючої бази даних Microsoft Access (Рис. 1).

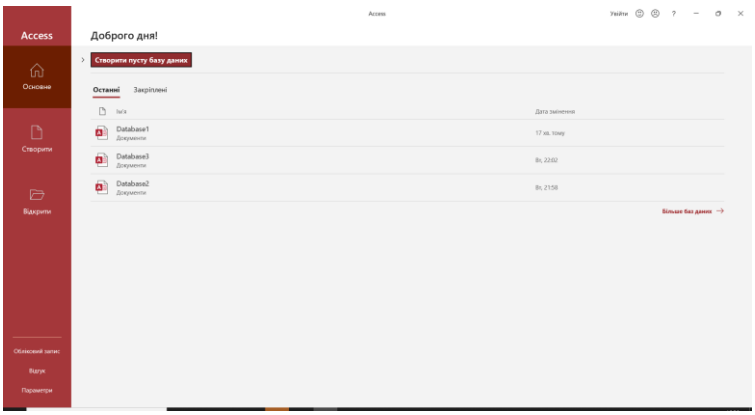


Рис.1. Діалогове вікно створення або відкриття бази даних

1. *Створення бази даних*

У результаті вибору кнопки **Створити Нову базу даних** створюється порожня база даних, в яку пізніше можуть бути поміщені таблиці, форми, звіти, запити і т.д. Створити базу даних можна також через головне меню **Файл**, вибравши **Пуста база даних** із **підменю Створити.** Відкривається вікно **Створення бази даних** для задання імені бази даних (рис. 2), в якому задаємо папку та ім’я файла (по замовчуванню назва та розширення файла Database1.accdb).

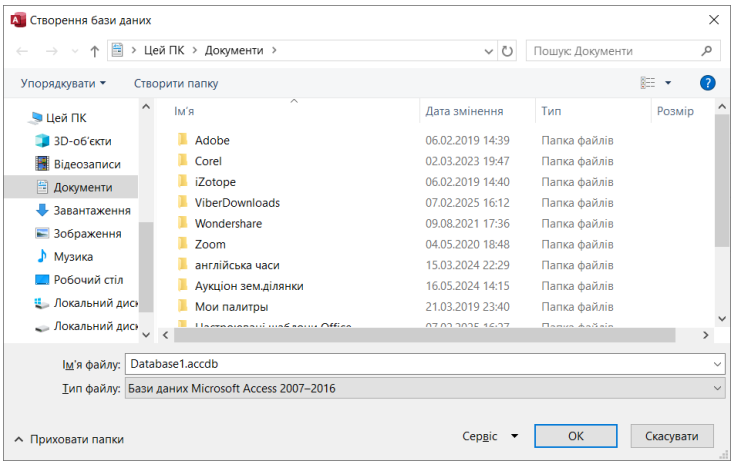


Рис. 2. Вікно вибору директорії та імені для нової бази даних

У результаті вибору кнопки ОК на екрані з’являється вікно бланка порожньої бази даних (рис. 3). У цьому вікні користувач може створювати нові таблиці, форми, запити та звіти.

Для зручності роботи в середовищі Microsoft Access передбачено панель інструментів (рис. 3*), яка містить основні кнопки для виконання найпоширеніших операцій — відкриття та збереження об’єктів, переходу між режимами, сортування, фільтрації, а також швидкого доступу до конструкторів таблиць, форм і запитів.

Панель інструментів полегшує навігацію в базі даних і дозволяє значно прискорити виконання базових дій, не звертаючись до головного меню програми.

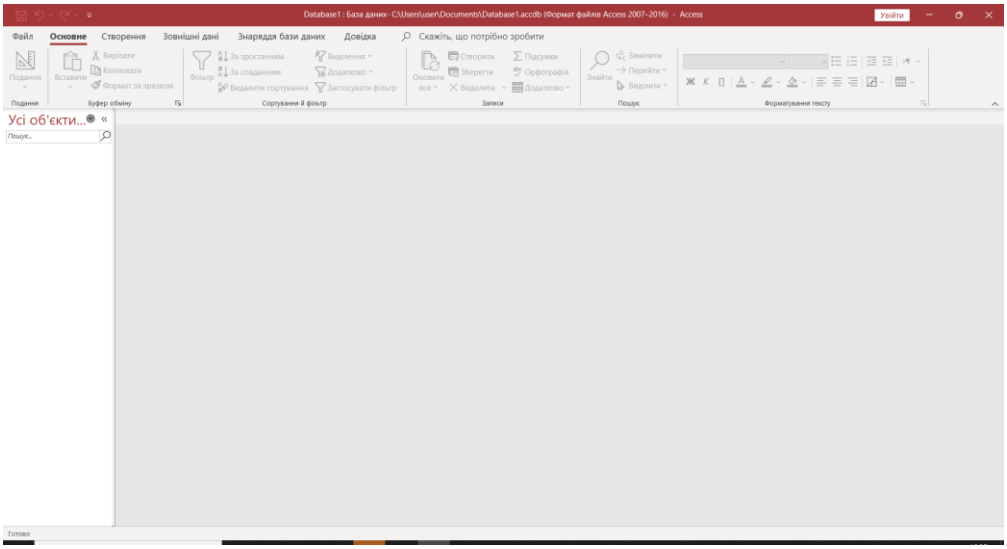


Рис. 3. Вікно порожньої бази даних

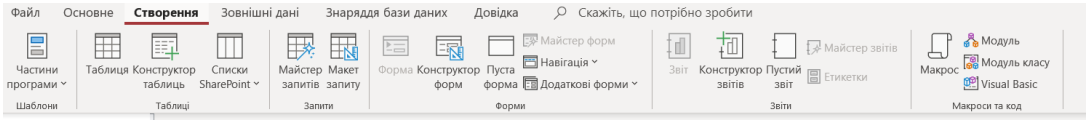
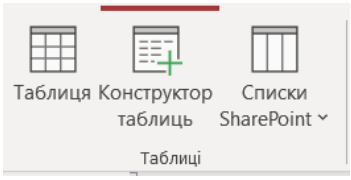


Рис.3*. Панель інструментів для створення бази даних

2. Створення структури таблиці

В головному меню обираємо пункт **Створення**, який складається із шести вкладинок : **Таблиці, Запити, Форми, Звіти, Макроси, Модулі.** Вкладинка **Таблиці** надає можливість створити таблицю бази даних двома способами: **Таблиця**(рис.4) і **Конструктор таблиць** (рис.4*).



Розглянемо можливі способи створення таблиці.

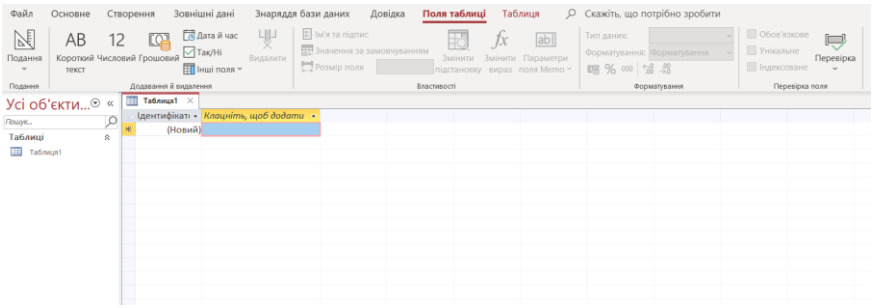


Рис. 4. Діалогове вікно створення способом **Таблиці**

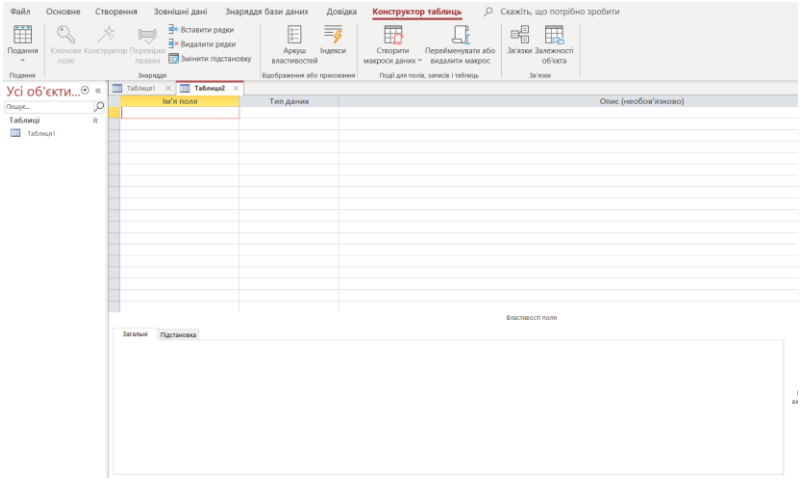


Рис. 4*. Діалогове вікно створення способом **Конструктор таблиць**

Вибираючи спосіб **Конструктор таблиць** переходимо до проектування нової таблиці за допомогою конструктора таблиць. У вікні конструктора таблиць (рис. 5) містяться такі області: панель інструментів вікна конструктора таблиць, область введення полів, область властивостей полів.

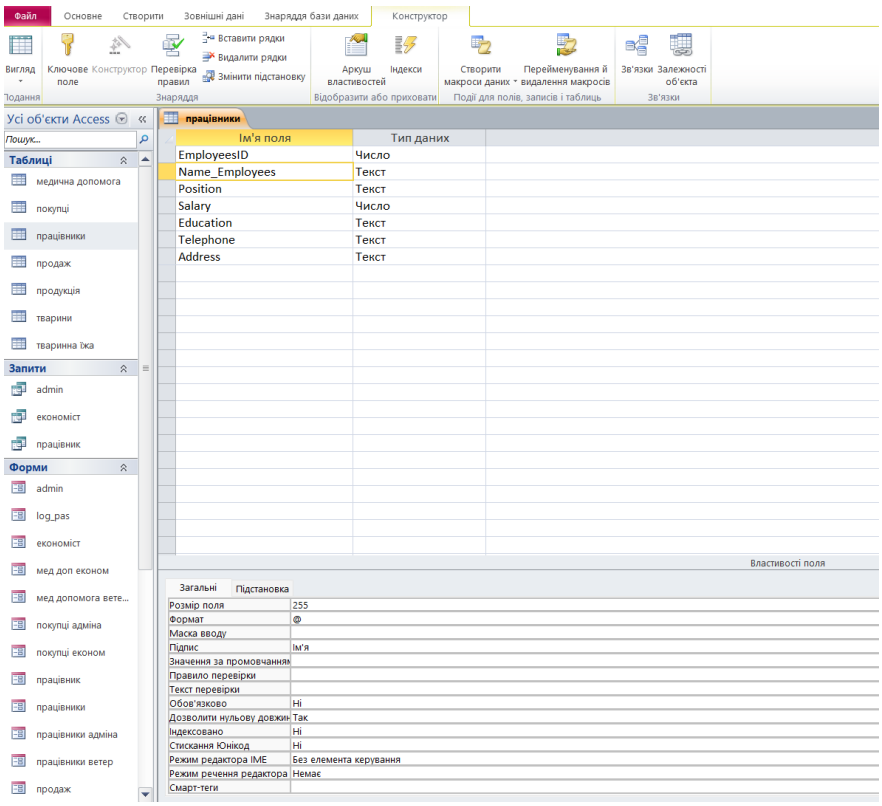


Рис. 5. Вікно конструктора таблиць

На панелі інструментів (рис. 6) вікна конструктора таблиць містяться кнопки, які полегшують створення структури нової таблиці.

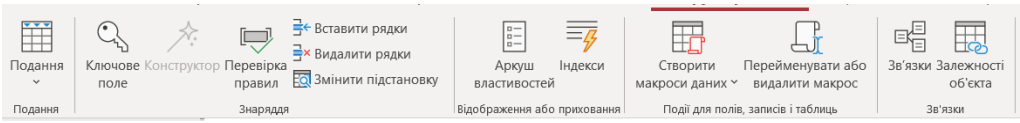


Рис. 6. Панель інструментів вікна конструктора таблиць

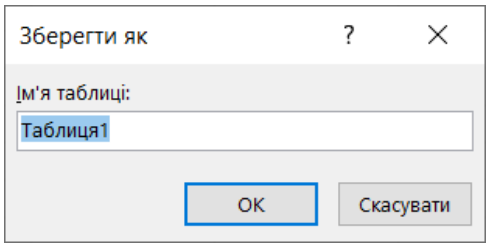
В області введення полів для кожного поля необхідно вказати його ім’я, тип, а також можна прокоментувати призначення поля в третій колонці **Опис**.

При створенні структури можна детально описати властивості кожного поля. Переміщатися між областю введення поля і областю властивостей поля можна за допомогою клавіші <F6> або мишки. Перерахуємо загальні властивості полів.

Розмір поля	Для тексту по замовчуванню 255
Формат	Змінює зовнішній вигляд даних після їх введення
Маска вводу	Використовується для введення даних в заздалегідь визначеному форматі
Підпис	Ярлик для полів, який замінює ім`я поля форми або звіту
Значення за замовчуванням	Значення, яке автоматично підставляється перед новим введенням даних у поле

Правило перевірки	Вираз, який обмежує значення, що можуть вводитися в полі.
Текст перевірки	Повідомлення про помилку, відображене в разі введення значення забороненого правилом поведінки
Обов'язково	Чи вимагати введення даних для цього поля
Дозволити нульову довжину	Чи дозволяти рядки з нульовим значенням цього поля
Індексовано	Вибір значення "Так-Без повторень" забороняє введення повторюваних значень в полі
Стискання Юнікод	Чи дозволяти стиснення Юнікод для цього поля
Режим редактора IME	Який режим редактора IME слід використовувати, коли фокус переходить до поля
Режим речення редактора IME	Який режим речення редактора IME слід використовувати, коли фокус переходить до поля
Вирівнювання тексту	Вирівнювання тексту в елементі керування

Після завершення створення структури таблиці необхідно закрити вікно конструктора таблиць. У цей момент програма Microsoft Access автоматично запропонує зберегти створену таблицю, відкривши діалогове вікно для введення її імені.

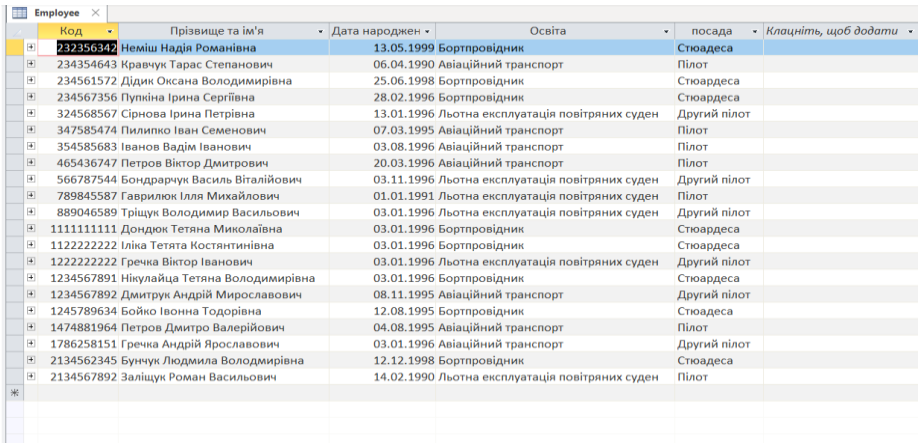


У полі введення користувач має вказати змістовне ім'я, яке відображатиме призначення таблиці (наприклад, Студенти, Предмети, Оцінки). Після підтвердження вибору натисканням кнопки ОК таблиця буде збережена у структурі бази даних і з'явиться в панелі навігації серед інших об'єктів.

Після введення імені система запитує, чи будемо задавати ключ. Встановлення ключа не є обов'язковим. Спосіб встановлення ключа описано в пункті 11.

2. *Введення даних у таблицю*

Для введення даних необхідно у вікні бланка бази даних (рис.3) зліва активізувати потрібну таблицю і натиснути кнопку “Відкрити” або подвійно клікнути на назві таблиці. Відкриється вікно для заповнення таблиці (рис.7). У цьому вікні дані можна як вводити, так і редагувати.



Employee

Код	Прізвище та ім'я	Дата народжен	Освіта	посада	Клацніть, щоб додати
232356342	Неміш Надія Романівна	13.05.1999	Бортпровідник	Стюардеса	
234354643	Кравчук Тарас Степанович	06.04.1990	Авіаційний транспорт	Пілот	
234561572	Дідик Оксана Володимирівна	25.06.1998	Бортпровідник	Стюардеса	
234567356	Пупкіна Ірина Сергіївна	28.02.1996	Бортпровідник	Стюардеса	
324568567	Сірнова Ірина Петрівна	13.01.1996	Льотна експлуатація повітряних суден	Другий пілот	
347585474	Пилипко Іван Семенович	07.03.1995	Авіаційний транспорт	Пілот	
354585683	Іванов Вадім Іванович	03.08.1996	Авіаційний транспорт	Пілот	
465436747	Петров Віктор Дмитрович	20.03.1996	Авіаційний транспорт	Пілот	
566787544	Бондрарчук Василь Віталійович	03.11.1996	Льотна експлуатація повітряних суден	Другий пілот	
789845587	Гаврилюк Ілля Михайлович	01.01.1991	Льотна експлуатація повітряних суден	Пілот	
889046589	Тріщук Володимир Васильович	03.01.1996	Льотна експлуатація повітряних суден	Другий пілот	
1111111111	Дондюк Тетяна Миколаївна	03.01.1996	Бортпровідник	Стюардеса	
1122222222	Іліка Тетята Костянтинівна	03.01.1996	Бортпровідник	Стюардеса	
1222222222	Гречка Віктор Іванович	03.01.1996	Льотна експлуатація повітряних суден	Другий пілот	
1234567891	Нікулайца Тетяна Володимирівна	03.01.1996	Бортпровідник	Стюардеса	
1234567892	Дмитрук Андрій Мирославович	08.11.1995	Авіаційний транспорт	Другий пілот	
1245789634	Бойко Івонна Тодорівна	12.08.1995	Бортпровідник	Стюадеса	
1474881964	Петров Дмитро Валерійович	04.08.1995	Авіаційний транспорт	Пілот	
1786258151	Гречка Андрій Ярославович	03.01.1996	Авіаційний транспорт	Другий пілот	
2134562345	Бунчук Людмила Володмирівна	12.12.1998	Бортпровідник	Стюадеса	
2134567892	Заліщук Роман Васильович	14.02.1990	Льотна експлуатація повітряних суден	Пілот	

Рис.7. Вікно таблиці в режимі заповнення

У програмі Microsoft Access тип поля «Вкладення» дає змогу зберігати в базі даних кілька файлів (зображення, документи, таблиці, презентації тощо) безпосередньо в записі таблиці.

У вікні вкладення об’єктів користувач може додавати, вилучати або переглядати прикріплені файли, що значно спрощує роботу з пов’язаними матеріалами в межах однієї бази.

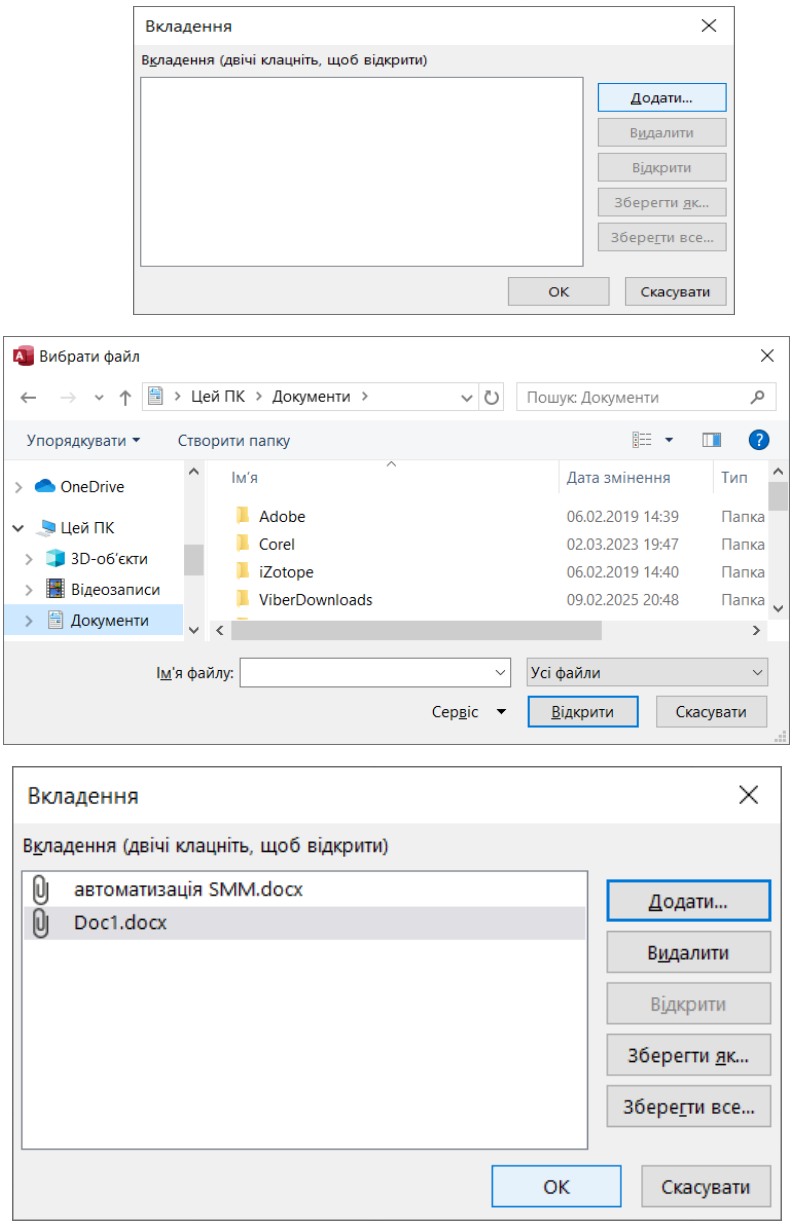


Рис.8. Вікна вкладення об’єктів в тип поля **Вкладення**

4.Закриття таблиці і бази даних

Після завершення введення, редагування та перевірки даних у таблиці необхідно закрити вікно з її вмістом. Для цього можна скористатися кнопкою **Закрити** у верхньому правому куті вікна або командою **Файл → Закрити**.

Наприклад, після роботи з таблицею **Employee** (рис. 7) її закриття повертає користувача у вікно бланка бази даних (рис. 3), де відображається список усіх об’єктів поточної бази — таблиць, форм, запитів і звітів. У цьому вікні можна перейти до інших елементів або виконати подальші дії з базою даних.

Якщо ж закрити вікно самої бази даних, тобто головне робоче середовище Access, — програма автоматично завершує роботу з поточною базою. Перед закриттям система, як правило, пропонує зберегти внесені зміни, щоб уникнути втрати оновлених даних.

Рекомендується завжди зберігати результати перед виходом, особливо після створення або модифікації структури таблиць, форм чи звітів. Це гарантує збереження актуального стану бази даних і полегшує подальшу роботу під час наступного відкриття.

5. Відкриття бази даних і таблиць бази даних

Відкрити створену раніше базу даних можна різними способами. При завантаженні Access з'являється діалогове вікно створення та відкриття бази даних (рис. 1). У цьому вікні необхідно вибрати кнопку **Відкрити** і із списку наведених нижче баз даних вибрати потрібну. Якщо в цьому списку її не має, то необхідно вибрати кнопку **Огляд** і у вікні **Відкрити** відшукати необхідну базу даних і натиснути кнопку **Відкрити**.

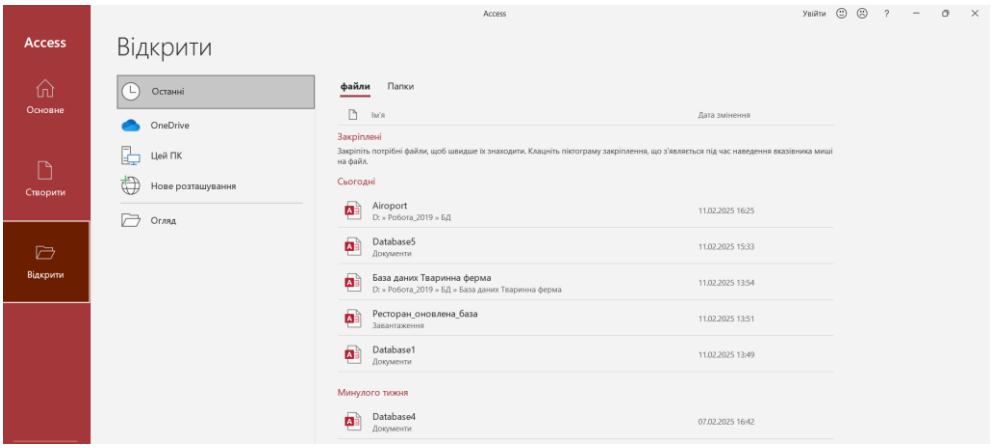


Рис.9. Вікно відкриття бази даних

Відкрити базу даних можна через головне меню, вибравши **Відкрити** із меню **Файл.**

В результаті відкриття бази даних з'являється вікно бланка бази даних (рис.3), в якому можна відкрити потрібну таблицю. Для цього вибираємо її і натискуємо кнопку **Відкрити**, або двічі натискуємо ліву клавішу мишки на вибраній таблиці.

6.Зміна структури таблиці бази даних

Для зміни структури таблиці відкритої бази даних необхідно натиснути кнопку **Конструктор** у вікні бланка бази даних (рис.3). Відкривається вікно конструктора таблиці (рис.5).

Щоб додати нове поле, курсор підводимо до поля, перед яким воно повинно бути, і виконуємо одну із таких дій:

– на клавіатурі натискаємо клавішу <Insert>;
– натискаємо праву клавішу мишки і із контекстного меню вибираємо команду **Додати рядки**;
– на панелі інструментів (рис. 6.) натискаємо кнопку **Додати рядки**.

Для вилучення поля необхідно встановити курсор на нього, виділити його лівою клавішею мишки і виконати одну із таких дій:

– на клавіатурі натиснути клавішу <Delete>;
– натиснути праву клавішу мишки і із контекстного меню вибрати команду **Видалити рядки**;
– на панелі інструментів (рис. 6) натиснути кнопку **Видалити рядки**.

7. Редагування, вилучення і добавлення нових записів таблиці

Для внесення змін до даних таблиці потрібно відкрити базу даних і перейти до вікна бланка бази даних (рис. 3). У списку об’єктів вибираємо потрібну таблицю та натискаємо кнопку Відкрити. У результаті відкриється вікно, у якому відображаються всі раніше введені записи.

У цьому вікні користувач може виконувати три основні дії з даними — редагувати, додавати та вилучати записи.

Редагування запису здійснюється безпосередньо у вікні таблиці. Для цього потрібно:

- Навести курсор у відповідну клітинку поля.
- Внести необхідні зміни до тексту або числового значення.

- Натиснути клавішу <Enter> або перейти до іншого поля — зміни зберігаються автоматично.
- Під час редагування Access контролює правильність введення даних відповідно до типів полів (наприклад, числовий, текстовий, дата/час тощо).

Щоб додати новий запис, потрібно:

- Перейти до останнього рядка таблиці, позначеного зірочкою (*).
- Ввести значення у відповідні поля нового запису.
- Після заповнення всіх полів натиснути <Enter> — запис буде автоматично додано до таблиці.
- Також новий запис можна вставити за допомогою команди **Основне → Записи → Створити**

Існує кілька способів вилучення записів із таблиці:

- Видалення одного запису:

Розташуйте курсор у будь-якому полі потрібного запису, а потім скористайтеся командою

Основне → Записи → Видалити або натисніть клавішу <Delete>.

- Видалення кількох записів:

Виділіть кілька рядків (утримуючи клавішу < Ctrl > або < Shift >) і натисніть <Delete>, або виберіть команду **Видалити → Видалити запис** на панелі інструментів.

Після цього з’явиться діалогове вікно підтвердження, у якому потрібно обрати **Так**, щоб остаточно вилучити записи з таблиці. Якщо натиснути **Ні**, дія скасовується, і дані залишаються без змін.

Після вилучення записів відновити їх неможливо, якщо перед цим не було створено резервної копії бази даних. Тому перед масовим редагуванням або видаленням рекомендується зберегти копію файлу бази (.accdb).

8. Перегляд таблиць

У результаті відкриття таблиці на екрані відображаються вся введена в неї інформація. Однак може виникнути потреба змінити вигляд даних. Можна розширити, звузити стовпці, перенести їх в інше місце, приховати, змінити висоту відображення запису, відображення сітки, шрифт відображення. Такі зміни впливають тільки на зовнішній вигляд таблиці, внутрішній порядок даних не змінюється. Зовнішній вигляд таблиці можна змінити в такий спосіб.

- *Для зміни ширини стовпця* необхідно встановити вказівник мишки на правій межі стовпця так, щоб він набув форми горизонтальної двонаправленої стрілки. Потім клацнути кнопкою мишки і перенести межу стовпця. Ширину стовпця можна змінити також, вибравши на панелі інструментів **Додатково** команду **Ширина поля**.
- *Для зміни висоти рядка* необхідно помістити вказівник мишки зліва між двома записами так, щоб він набув форми вертикальної двонаправленої стрілки. Клацнути кнопкою мишки і перенести межу рядка. Теж саме можна виконати, на панелі інструментів **Додатково** команду **Висота рядка**.
- *Для переміщення стовпця в інше місце* необхідно виділити його, клацнувши мишкою на назві, клацнути кнопкою мишки ще раз і перенести стовпець у нове місце.
- *Щоб приховати стовпець,* необхідно виділити його, а потім вибравши на панелі інструментів **Додатково** команду **Приховати поля**.
- *Для відображення прихованого стовпця,* потрібно на панелі інструментів **Додатково** вибрати команду **Приховати поля**.
- *Для зміни шрифтів та їх налаштувань* необхідно використовувати команди на панелі налаштувань **Форматування тексту** (рис.10).

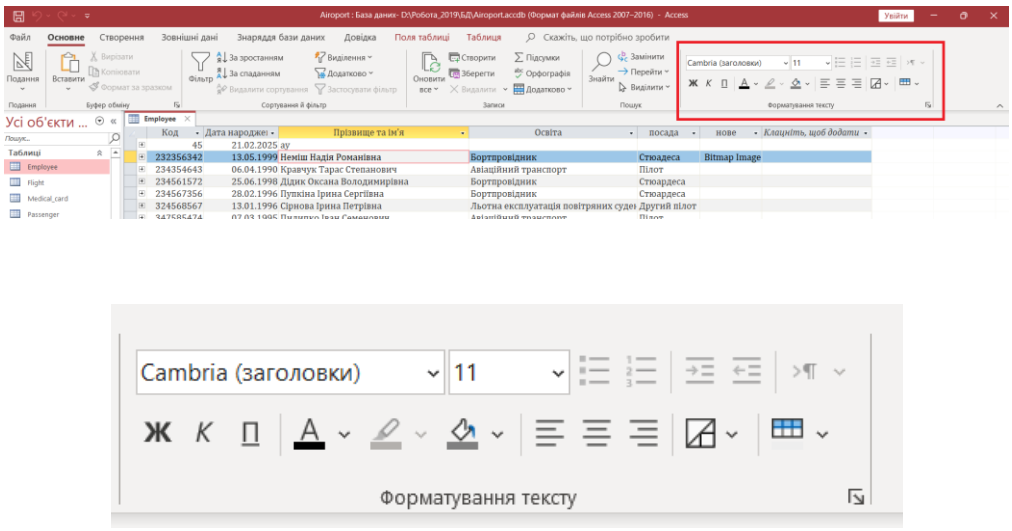


Рис. 10. Форматування тексту

Можна налаштувати шрифт, його розмір, задати підкреслення, змінити колір та інші параметри. Після зміни зовнішнього вигляду таблицю закриваємо.

9. Пошук даних

Для пошуку запису необхідно виконати такі дії:

- Відкрити таблицю, в якій потрібно знайти запис, а потім клацнути мишкою в будь-якому місці поля, по якому шукаємо запис.
- Виконати команду **Знайти** на панелі налаштувань, з'явиться діалогове вікно **Пошук і заміна** (рис.11)

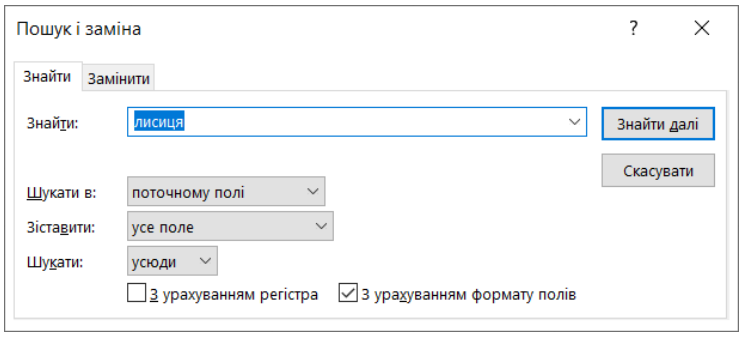


Рис.11. Діалогове вікно **Знайти**

- У полі **Знайти** ввести значення поля, яке потрібно знайти.
- У списку **Зіставити** необхідно вибрати **будь-яку частину поля** для пошуку підрядка, **усе поле** для пошуку запису, в якому вміст поля зберігається з вмістом рядка пошуку або **Початок поля** для пошуку за початком вмісту поля.

- Якщо залишити прапорець опції **Поточному полі**, то пошук відбуватиметься по заданому полю. Прапорець опції **З урахуванням регістра** вказує на те, чи розрізняти при пошуку малі й великі букви. Якщо встановити прапорець на опцію **З урахуванням формату полів,** то пошук буде здійснюватися на основі формату відображення поля.

10. Сортування записів

Іноді необхідно здійснити сортування записів у певній послідовності. Для вибору поля сортування досить помістити курсор у довільний його запис. Після цього клацнути по кнопці сортування на панелі інструментів (за зростанням, за спаданням).

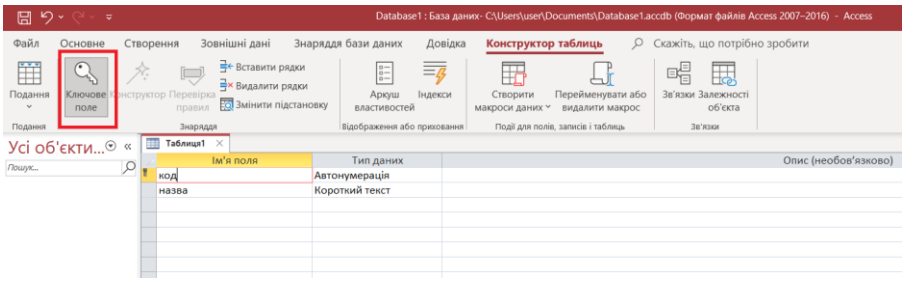
У результаті дані будуть відображені у відсортованому порядку.

Якщо необхідно відсортувати дані за декількома полями, то виділяємо декілька стовпців і натискаємо відповідну кнопку. В результаті записи відсортуються спочатку за першим стовпцем, потім – за другим і т.д..

11. Створення первинних ключів

Кожна таблиця може мати первинний ключ. Він ідентифікує записи й дозволяє відрізнити один запис від іншого. Первинний ключ складається із одного або декількох полів.

Для створення первинного ключа використовують кнопку панелі інструментів



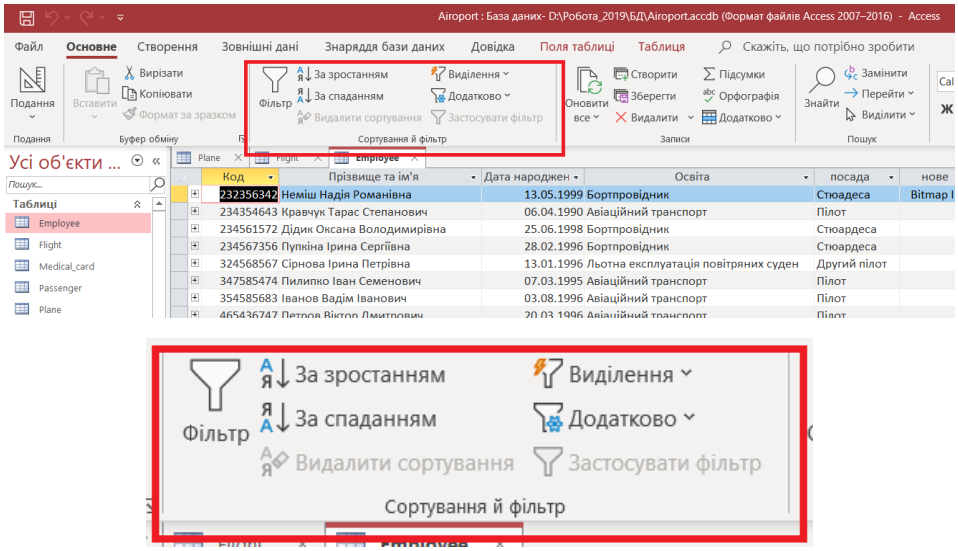
Щоб виділити одне поле в цьому вікні, необхідно клацнути лівою клавішею мишки на вказівнику активного поля. Для виділення декількох полів необхідно додатково тримати натиснутою клавішу <Ctrl>. Для того, щоб виділені поля складали первинний ключ, необхідно натиснути кнопку

Ключове поле на панелі інструментів або натиснути праву клавішу мишки і з контекстного меню вибрати команду **Ключове поле**. В результаті у вказівнику поля первинного ключа з'явиться символ ключика.

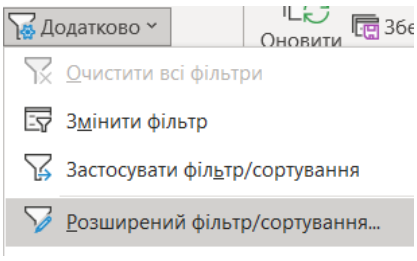
12. Фільтрація даних

Бази даних можуть містити велику кількість записів. Типовим завданням обробки даних є задача пошуку записів, які задовольняють певну умову. Для пошуку потрібних записів використовують фільтри і запити. Фільтри застосовують у випадку нескладних умов пошуку.

Доступ до команд роботи з фільтрами можна отримати на панелі інструментів:



- Фільтр;
- Виділення;
- Додатково (Змінити фільтр, Застосувати фільтр/ сортування, Розширений фільтр/сортування);



Під час експериментів необхідно застосувати команду **Очистити всі фільтри** із пункту **Додатково**, щоб повернути початкове відображення таблиці.

Для пошуку даних користувач будує умови – прості й складені.

Прості умови – це числа, тексти, вирази, математичні співвідношення, наприклад: 2; «Студент»; >3; =4; <>5; <=date() – усі дати до сьогодні включно; Like (A*) – прізвища, які починаються на А, Like (*02) – слова, які мають закінчення 02.

Складені умови – це умови , побудовані з простих за допомогою логічних операцій Not (не), And(і), Or(або), between (між).

Наприклад, Not 2; Between 3 and 4; between date()-30 and date() – дати за минулі 30 днів від сьогоднішньої; between 21.01.2025 and 20.02.2025 – між двома датами.

Для задання простого фільтра необхідно, наприклад, виконати команду **Змінити фільтр** із **Додатково**. У результаті на екрані з'явиться вікно із рядком полів таблиці. Обираємо необхідне поле і через кнопку розкриття списку встановлюємо значення поля, за яким будемо фільтрувати записи або задавати умову. Далі виконуємо команду **Застосувати фільтр** на панелі інструментів**.**

Простий фільтр можна задати, використовуючи також опцію **Виділення** на панелі інструментів. Для цього необхідно вибрати довільний запис, виділити значення поля, за яким відбуватиметься фільтрація, і виконати команду **Виділення**, а далі вказати опцію (Дорівнює "Значення поля", Не дорівнює "Значення поля", Містить "Значення поля", Не містить "Значення поля").

Для задання розширеного фільтра потрібно вибрати опцію **Розширений фільтр/сортування** команди **Додатково.** З'явиться діалогове вікно побудови розширеного фільтра (рис. 12).

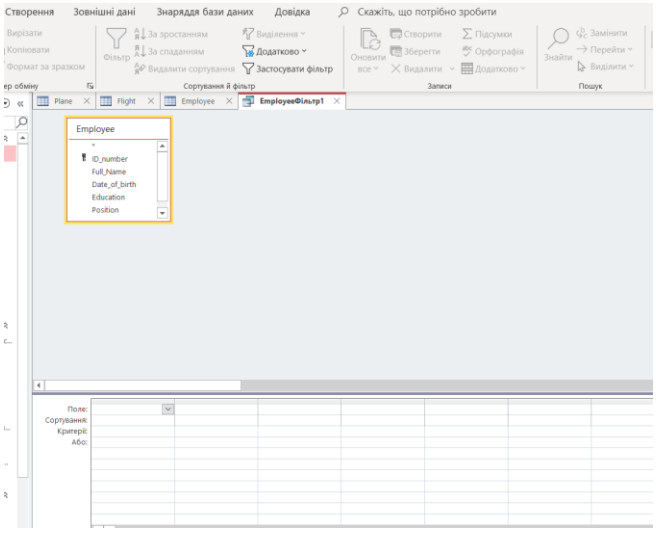
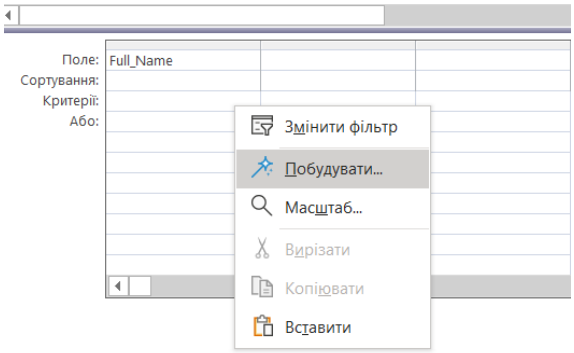


Рис.12. Вікно побудови розширеного фільтра

У верхній частині вікна із списку полів таблиці вибирають потрібні й перетягують їх у рядок **Поле**. Інший спосіб вибору полів у рядку **Поле** – вказати імена полів, по яких задається умова. В рядку **Критерій** задати умови, які накладаються на ці поля. Тут можна також вказати спосіб сортування (рядок **Сортування**). Після задання фільтра виконуємо команду **Застосувати фільтр.**

При заданні фільтрів можна виконувати над полями дії. Для цього у вікні побудови розширеного фільтра (рис.12) у рядках **Поле** або **Критерій** клацнути правою клавішею мишки і з контекстного меню вибрати команду **Побудувати**.



У результаті відкривається вікно **Побудовника виразів** (рис.13).

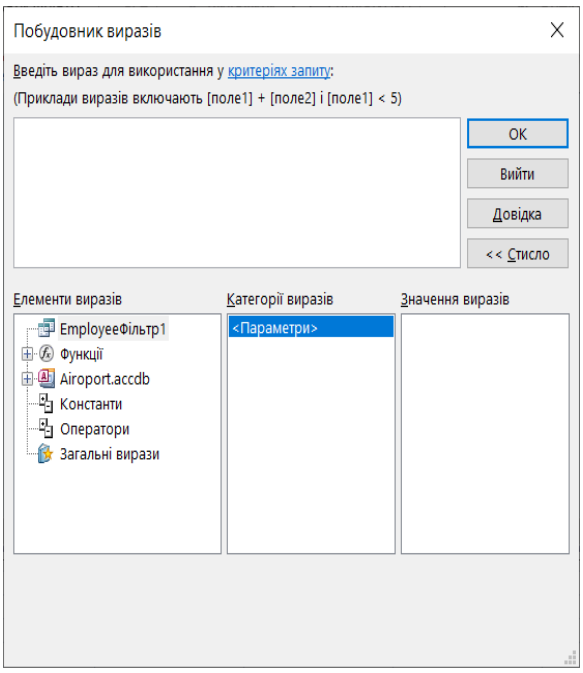


Рис.13. Діалогове вікно **Побудовник виразів**

Вираз може складатись із імен полів таблиць, констант, функцій чи операторів. Вираз можна писати безпосередньо у вікні **Побудовник виразів** або будувати за елементами об’єктів чи за їх значеннями, які виводяться у нижній частині вікна.

Увага! В кінці кожної вправи необхідно застосувати команду **Очистити всі фільтри** із пункту **Додатково**, щоб повернути початкове зображення таблиці.

Лабораторна робота №3

Створення запитів у середовищі MS Access

ЗАВДАННЯ

I. Використовуючи вбудовані засоби MS Access, реалізувати запити різної складності:

1. Запит на вибірку із простими умовами.
2. Запит на вибірку із складеними умовами
3. Запит із обчислювальним полем
4. Запит на вибірку із групуванням.
5. Параметричний запит
6. Перехресний запит (з використанням відповідного майстра та без нього)
7. Запити на зміни:
   - для створення таблиці;
   - для оновлення даних;
   - для додавання даних;
   - для знищення даних.

МЕТОДИЧНІ ВКАЗІВКИ

1. Відкрити вікно для створення запиту
2. Відібрати поля у запит
3. Відсортувати записи
4. Відібрати записи
5. Зберегти запит
6. Виконати запит

Запит – це засіб відшукання записів, перетворення таблиць і створення на їхній основі нових.

Розрізняють декілька типів запитів. Найпростішим є звичайний запит (*запит на вибірку*), який відображає на екрані вибрані з таблиці записи. Ці запити не змінюють таблиці.

Для створення нової таблиці, що міститиме вибрані записи, внесення змін у таблицю (доповнення, оновлення, вилучення чи архівування записів, створення обчислювального поля тощо), використовують запити на перетворення (на внесення змін, на виконання дії з таблицею).

Тип запиту можна змінювати командами на панелі інструментів вкладки **Запити**.

Загальна схема роботи з запитами на внесення змін така:

1) виконують звичайний запит на вибірку записів;
2) змінити тип запиту, наприклад на створення нової таблиці;
3) запускають запит на виконання – буде створена нова таблиця з відібраними записами.

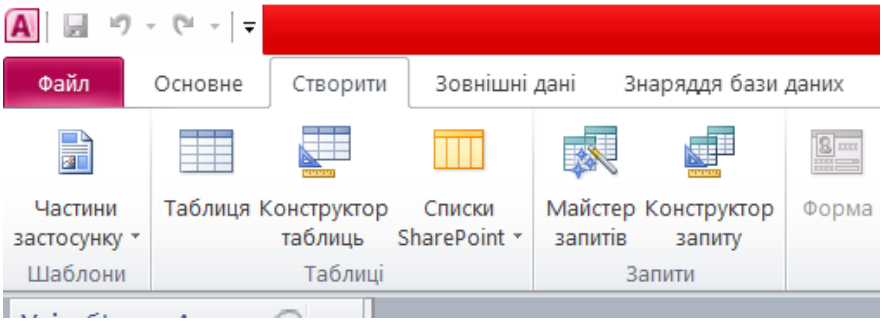
Запит створюють вручну в режимі конструктора або за допомогою майстра запиту на базі деякої таблиці чи декількох таблиць, яку/які додають до запиту.

Після створення таблиць і введення в них даних можна приступити до створення запиту.

1. Відкриття вікна для створення запиту

Щоб створити запит необхідно виконати дії у такій послідовності:

1) у вікні бланка бази даних (рис.3) клікнути на вкладці **Запити**



2) якщо обираємо **Майстер запитів**, тоді з’явиться діалогове вікно **Новий запит** (рис.14) з чотирма способами побудови запиту;

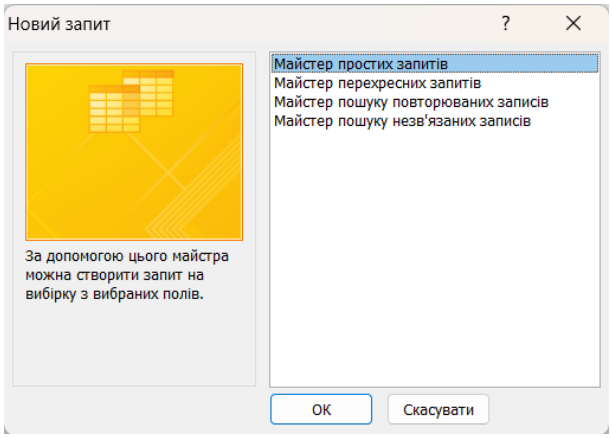


Рис. 14. Вікно **Новий запит**

3) якщо обираємо **Конструктор запитів**, тоді відкривається два вікна: діалогове вікно **Відображення таблиці** у запит і конструктор запитів (рис.15)

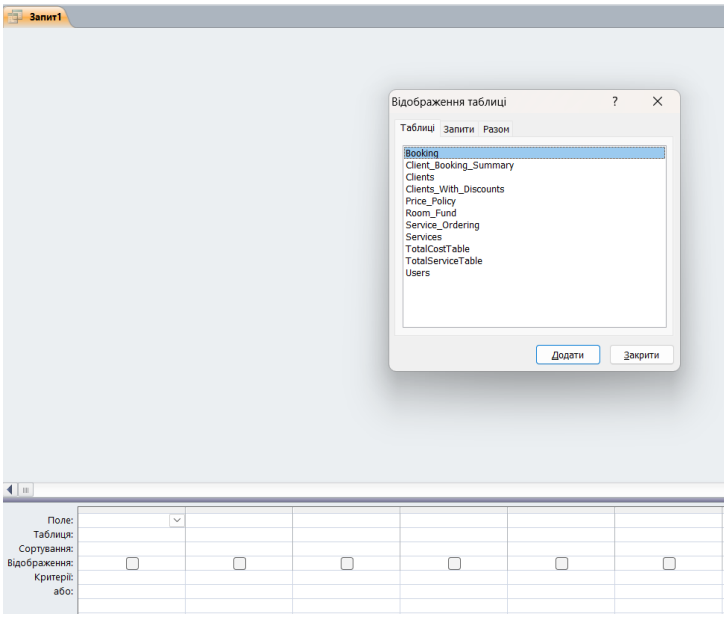


Рис. 15. Діалогове вікно добавлення таблиці у конструктор запитів

У вікні **Відображення таблиці** відображаються всі таблиці й запити бази даних.

Вибираємо потрібну таблицю і натискаємо кнопку **Додати,** а потім **Закрити**. У результаті з’явиться вікно **Запит 1** (рис.16).

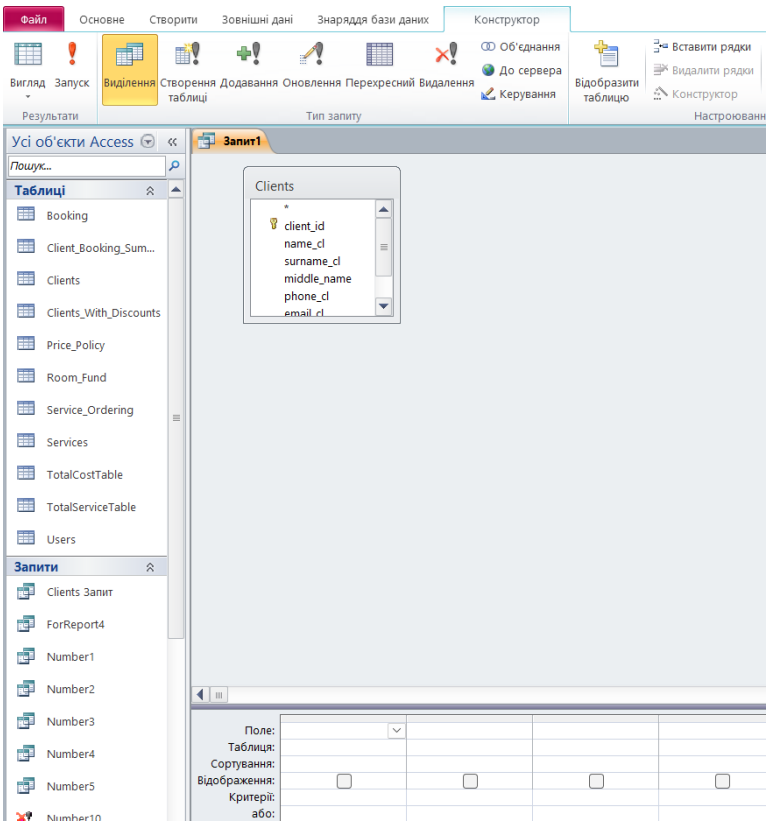
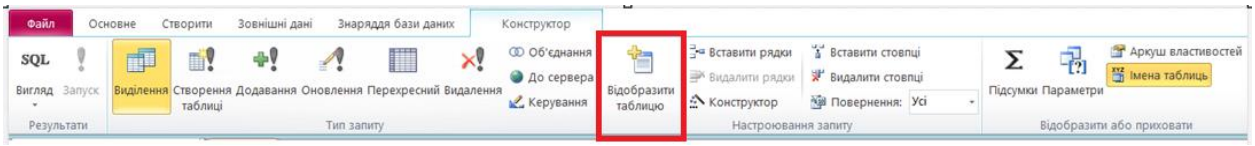


Рис.16. Вікно конструктора запиту

Вікно конструктора запиту має режими вигляду: **Подання таблиці**  і **Конструктор**  .

У режимі **Конструктор** створюється запит, а в режимі **Подання таблиці** виводиться результуючий набір даних запиту.

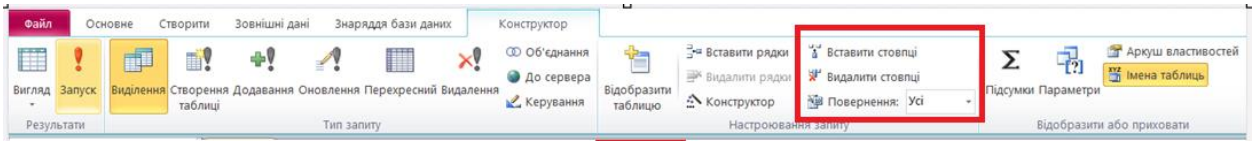
Викликати діалогове вікно **Відображення таблиці** для добавлення інших таблиць у вікно конструктора запитів можна за допомогою команди **Відобразити таблицю** із панелі інструментів або клікнути правою клавішею мишки у верхній частині вікна конструктора запитів і вибрати з контекстного меню Відобразити таблицю.



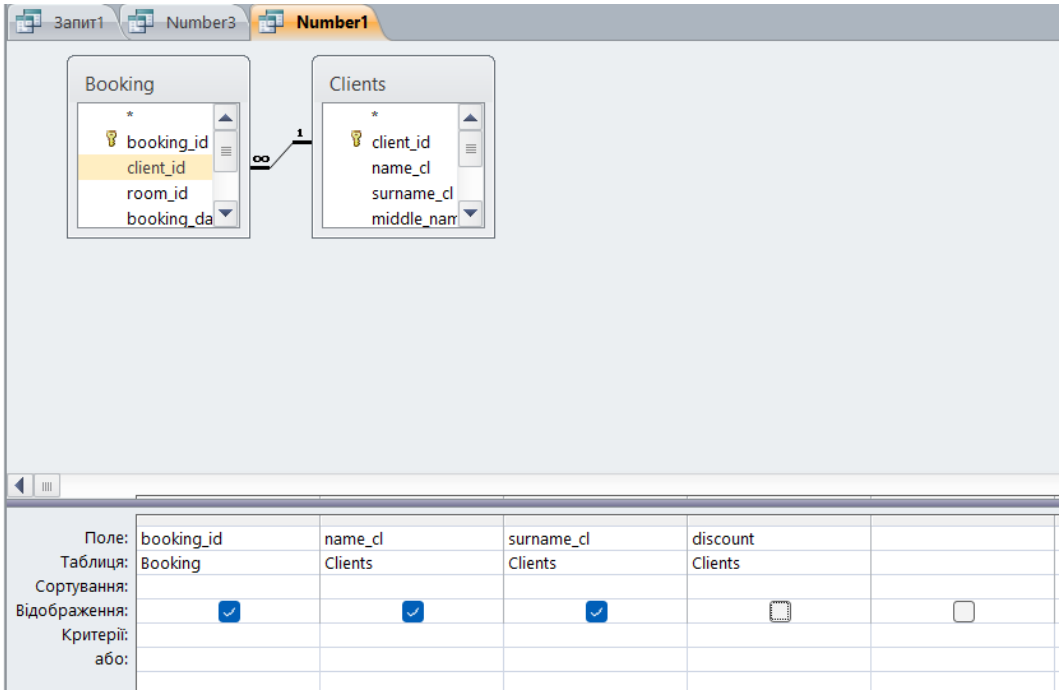
2. Відбір полів у запит

Для добавлення поля в запитах можна двічі клацнути мишкою на його імені в таблиці або перетягнути поле мишкою з відповідної таблиці в рядок **Поле.**

Для вилучення одного поля із рядка необхідно виділити його, а потім натиснути клавішу <Delete> або скористатись командою **Видалити стовпці** із панелі інструментів



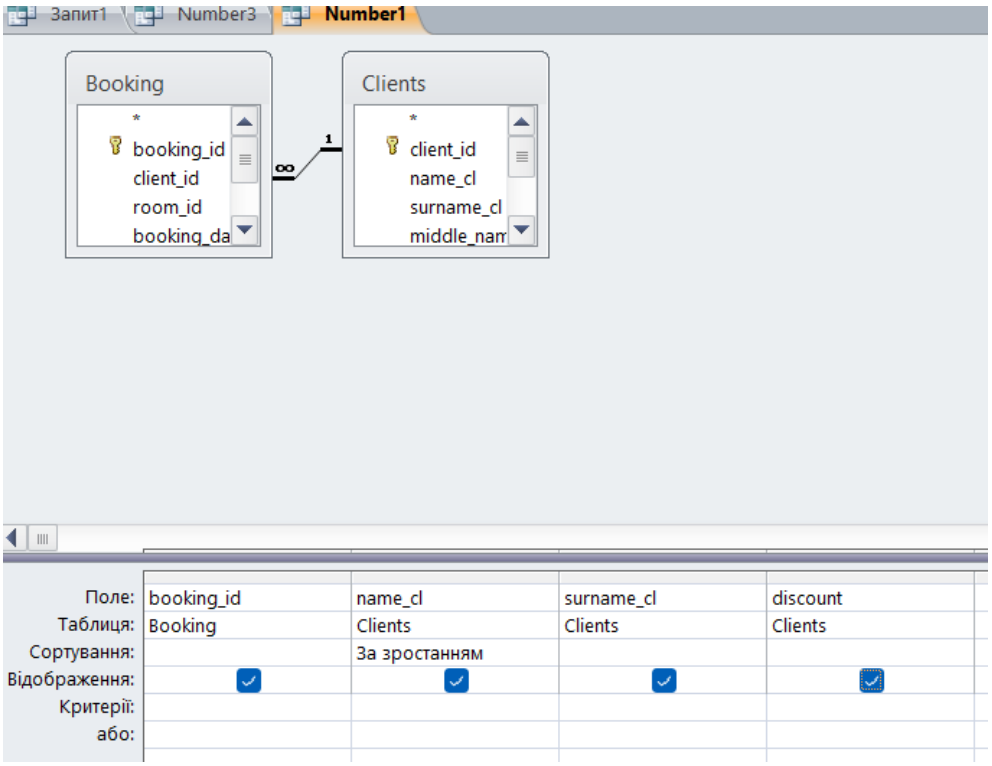
У рядку властивості **Відображення** необхідно встановити прапорці на полях, які будуть виводитись на екран при виконанні запиту.



За замовчуванням прапорці встановлені на всіх полях.

3. Сортування записів

Для сортування записів по певному полю необхідно встановити курсор у відповідному полі і клацнути на комірці властивості **Сортування**, а потім задати тип сортування (за зростанням, за спаданням). Тут є можливість сортування за декількома полями.



4. Відбір записів

Відбір записів у Microsoft Access — це процес вибору з таблиці лише тих даних, які відповідають певним умовам. Такий механізм дозволяє швидко знаходити потрібну інформацію без необхідності переглядати всю таблицю.

Для цього використовують умови відбору (фільтри), що задаються в запитах або безпосередньо під час роботи з таблицею чи формою.

Умови відбору формуються в рядку властивості **Критерій** у бланку запиту. Саме тут визначаються значення або логічні вирази, за якими Access буде шукати потрібні записи.

Критерії можуть включати:

- числові умови: >100, <50, Between 10 And 20;
- текстові умови: "Київ", "К*" (усі записи, що починаються з “К”);
- умови за датою: #01.01.2025#, Between #01.01.2025# And #31.12.2025#;
- логічні вирази: AND, OR, NOT для поєднання кількох умов.

Наприклад:

1. Відібрати всіх працівників, у яких посада = "інженер".

2. Знайти студентів, які мають середній бал більше 90.

3. Показати записи, дата прийому на роботу пізніше 01.01.2020.

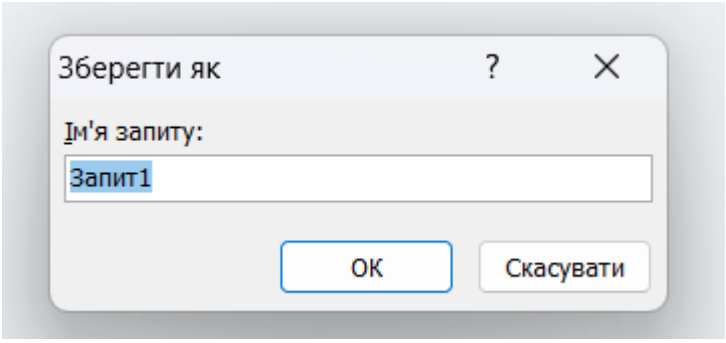
Вирази будуються як і відповідні вирази при фільтрації даних (дивись вказівки до лабораторної роботи №2, пункт 12).

5.Збереження запиту

Після завершення конструювання запиту його необхідно зберегти. Це можна зробити такими способами:

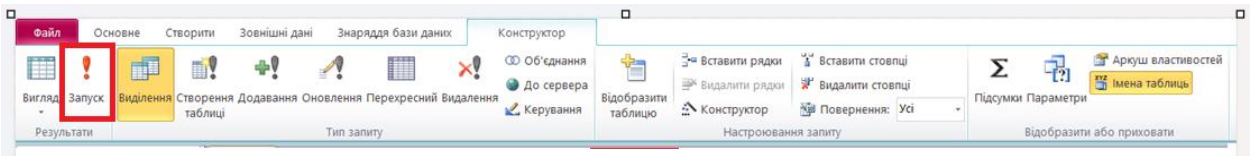
- вибрати команду **Закрити** із меню **Файл**;
- на панелі інструментів клацнути по кнопці **Закрити**;
- закрити вікно конструктора запитів і зберегти запит.

Всі ці методи приводять до появи діалогового вікна задання імені запиту. Задаємо ім'я і повертаємось у вікно бланка бази даних.



6. Виконання запиту

Для виведення результату запиту необхідно у вкладинці **Конструктор** натиснути піктограму **Запуск**

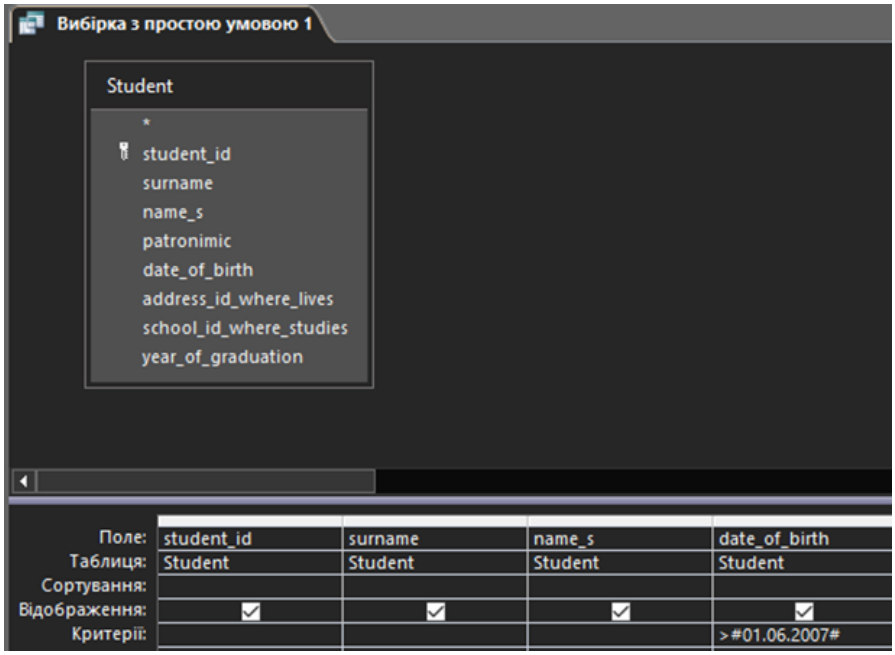


Також вивести результату запиту можна у вікні бланка бази даних (рис.3) вибрати вкладинку **Запити**. У результаті буде виведено список запитів, після чого вибираємо потрібний і натискаємо кнопку **Відкрити** або двічі натискаємо ліву клавішу мишки на імені запиту.

Розглянемо приклади запитів різної складності, згідно завдань лабораторної роботи.

Тип запиту: Запит на вибірку із простою умовою

Приклад: Вивести список учнів, які народились після 1 червня 2007 року.



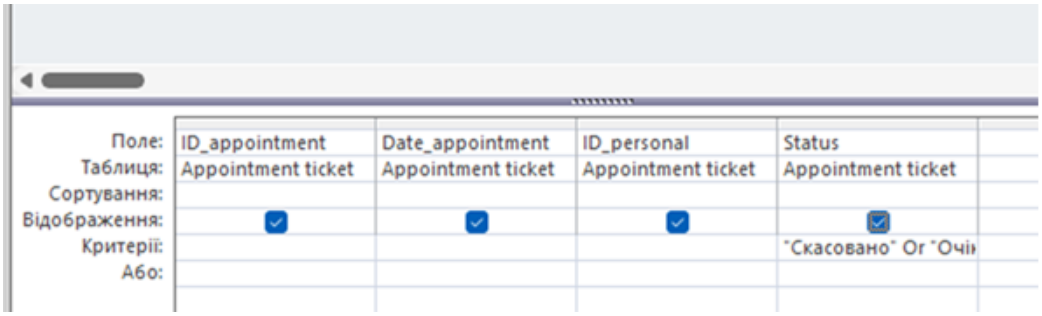
Результат:

Вибірка з простою умовою 1

ID студента	Прізвище	Ім'я	Дата народження
15	Кіндор	Володимир	17.07.2007
18	Козьма	Іван	30.11.2007
27	Пастор	Андрій	03.10.2007
28	Пономаренко	Ірина	13.03.2008
29	Попова	Ганна	13.04.2008
30	Романенко	Іван	27.05.2008
31	Рудь	Ігнат	11.09.2008
33	Севрюкова	Валентина	16.07.2008
34	Семененко	Поліна	24.01.2008
35	Сергієнко	Василь	25.05.2008
36	Сулим	Володимир	14.08.2008
37	Терлецький	Василь	29.10.2008
38	Тимченко	Ганна	16.06.2008
39	Турянський	Юлія	29.01.2008
41	Шевчук	Валерій	08.07.2008
42	Шудров	Андрій	22.04.2008
56	Кузьміна	Ніна	21.07.2007

Тип запиту: Запит на вибірку зі складеною умовою

Приклад: Знайти всі прийоми, які мають статус "Скасовано" АБО "Очікується"

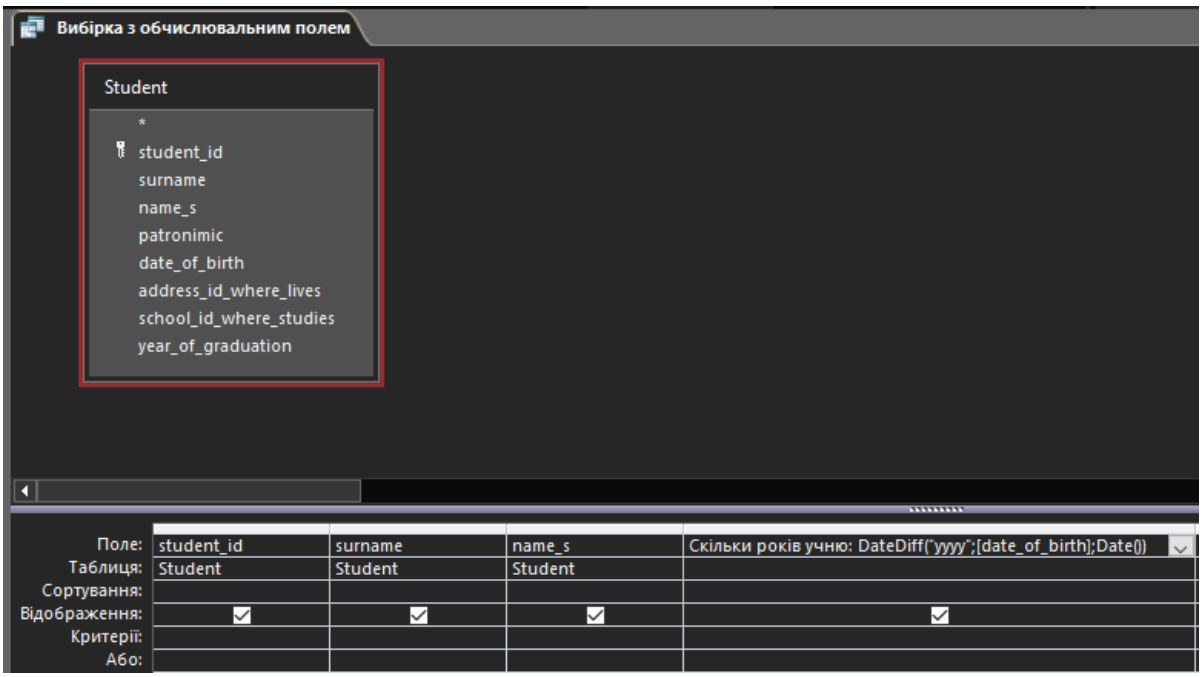


Результат:

Номер талс	Дата прийому	Номер лікаря	Статус
12	10.10.2025	Сидоренко	Очікується
15	10.10.2025	Сидоренко	Очікується
16	10.10.2025	Ткачук	Очікується
18	10.10.2025	Сидоренко	Очікується
21	10.10.2025	Романюк	Очікується
22	10.10.2025	Сидоренко	Очікується
23	10.10.2025	Ткачук	Очікується
26	10.10.2025	Паламарчук	Очікується
27	10.10.2025	Ткачук	Скасовано
28	10.10.2025	Сидоренко	Очікується
29	13.10.2025	Бондаренко	Очікується
32	14.10.2025	Бондаренко	Скасовано
33	15.10.2025	Сидоренко	Очікується
35	13.10.2025	Бондаренко	Очікується
(Новий)			

Тип запиту: Запит на вибірку з обчислювальним полем

Приклад1: Порахувати скільки років кожному учню.



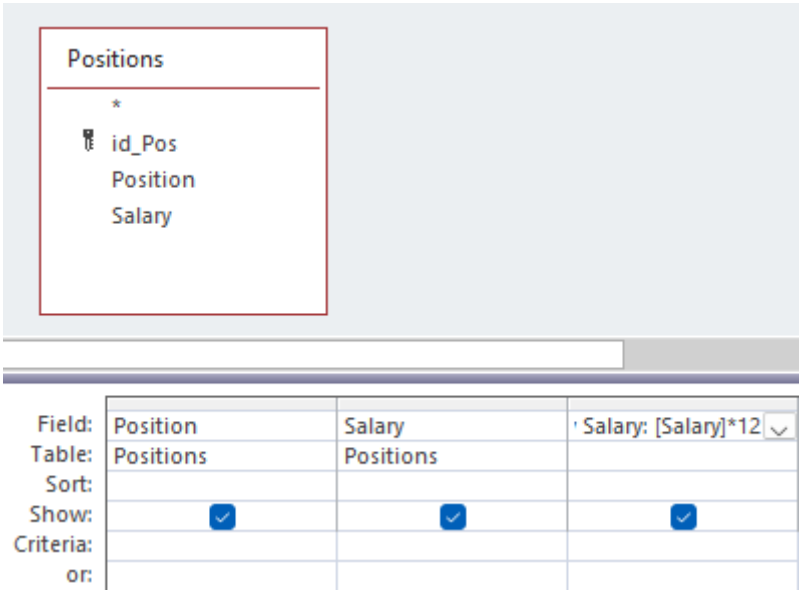
Результат:

Вибірка з обчислювальним полем

ID студента	Прізвище	Ім'я	Скільки років учню
1	Абрамюк	Ганна	19
2	Біленький	Андрій	19
3	Білоус	Галина	19
4	Бондаренко	Людмила	19
5	Войцехівський	Василь	19
6	Гаврилюк	Андрій	19
7	Гангал	Остап	19
8	Гончар	Іван	19
9	Гудзь	Ганна	19
10	Гуцул	Ірина	19
11	Дідів	Андрій	19
12	Жук	Іван	19
13	Журахівський	Павло	19
14	Кириленко	Іван	19
15	Кіндор	Володимир	18
16	Коваль	Андрій	18
17	Когут	Наталія	18
18	Козьма	Іван	18
19	Кузема	Василь	18
20	Кузьменко	Іван	18
21	Лермонтова	Інна	18
22	Лисенко	Ірина	18
23	Масюкевич	Василь	18
24	Мельник	Вадим	18
25	Мудряк	Валентина	18
26	Папущак	Іван	18
27	Пастор	Андрій	18
28	Пономаренко	Ірина	17
29	Попова	Ганна	17

Запис: 1 з 150 Без фільтра Пошук

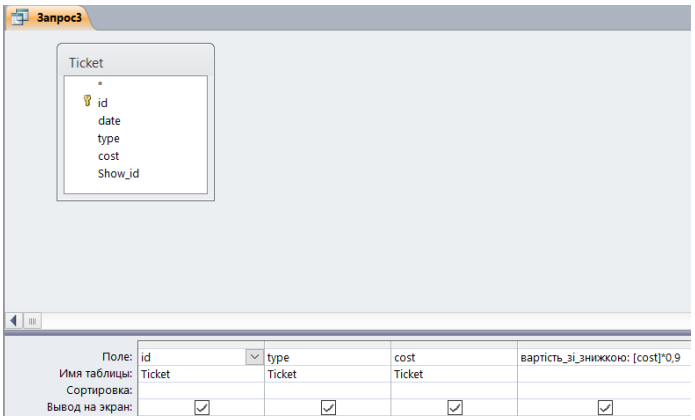
Приклад2: Виведемо річну зарплату, створивши нову колонку, помноживши зарплату на 12.



Результат:

Посада	ЗП	Yearly Salar
CEO	0,00 ₴	0
Security	30 000,00 ₴	360000
Curator	10 000,00 ₴	120000
Maintenance	8 500,00 ₴	102000
Marketing Specialist	14 500,00 ₴	174000

Приклад3: Вивести інформацію про квитки, додаючи поле з вартістю квитка зі знижкою 10%.



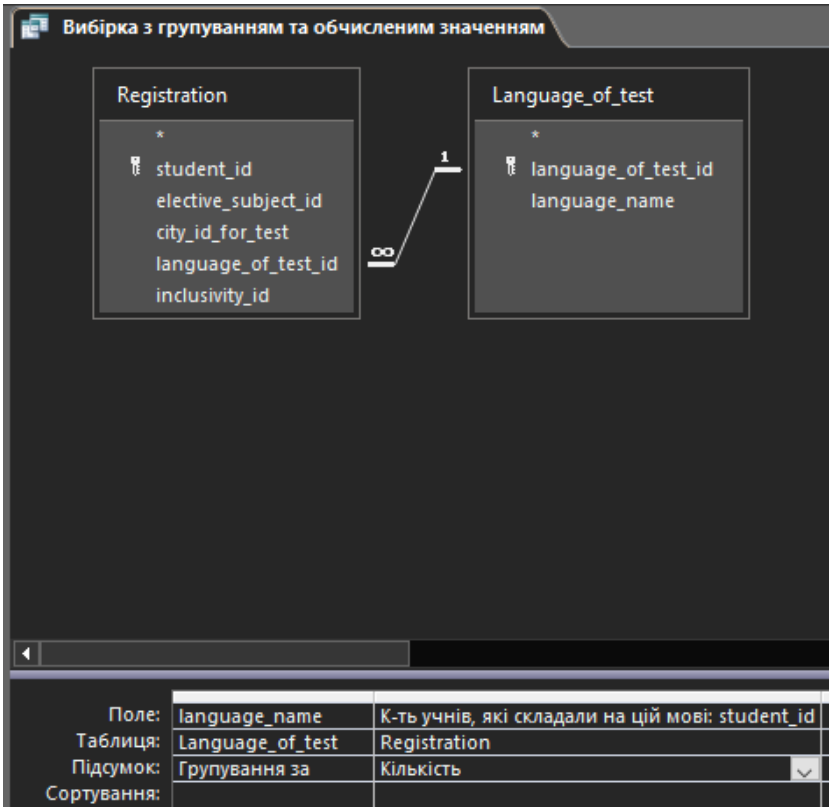
Результат:

Запрос3

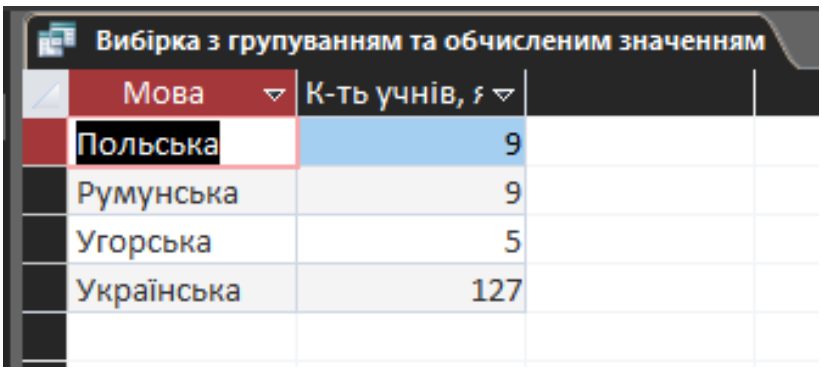
id	тип	вартість	вартість_зі_знижкою
1	Дорослий	150	135
2	Студентський	80	72
3	Дорослий	150	135
4	Пенсійний	50	45
5	Дитячий	0	0
6	Дорослий	150	135
7	Студентський	80	72
8	Дорослий	150	135
9	Дорослий	150	135
10	Пенсійний	50	45
11	Дорослий	150	135
12	Студентський	80	72
13	Дорослий	150	135
14	Дорослий	150	135
15	Пенсійний	50	45
16	Дитячий	0	0
17	Дорослий	150	135
18	Студентський	80	72
19	Дорослий	150	135
20	Дорослий	150	135

Тип запиту: Запит на вибірку із групуванням та обчисленим значенням

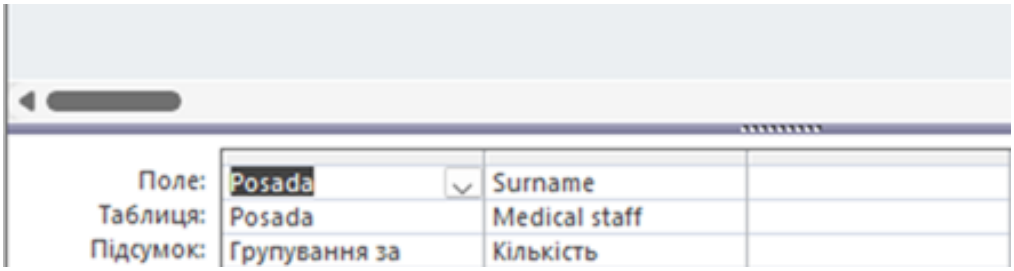
Приклад1: Для кожної мови порахувати, скільки учнів складало тест.



Результат:



Приклад2: Порахувати скільки працівників кожної посади.

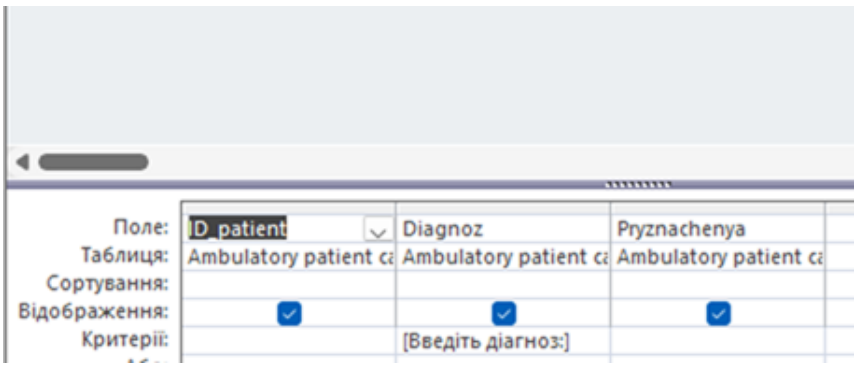


Результат:

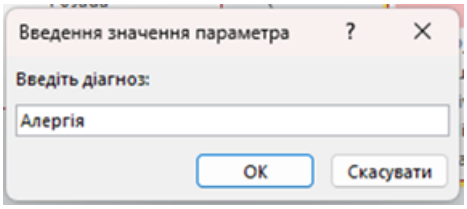
Посада	Кількість3Surname
Завідувач відділення	4
Лаборант	1
Лікар	4
Медична сестра	1
Реєстратор	2

Тип запиту: Запит на вибірку із параметром

Приклад: Показати всіх пацієнтів, яким поставили діагноз, що його введе користувач.



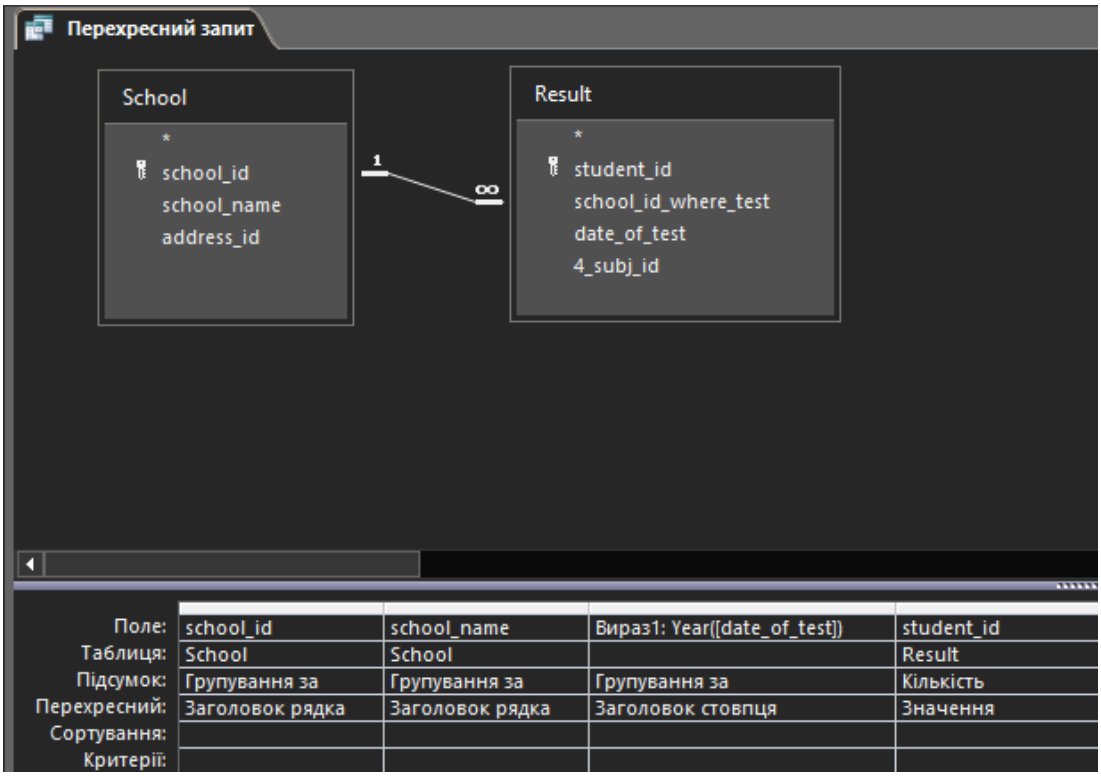
Результат:



Номер пацієнта	Діагноз	Призначення
Тарасюк	Алергія	Супрастин
Орлюк	Алергія	Супрастин
Горбулько	Алергія	Супрастин

Тип запиту: Перехресний запит

Приклад: Для кожної школи показати, скільки учнів складало тест в 2023, 2024 та 2025 роках.



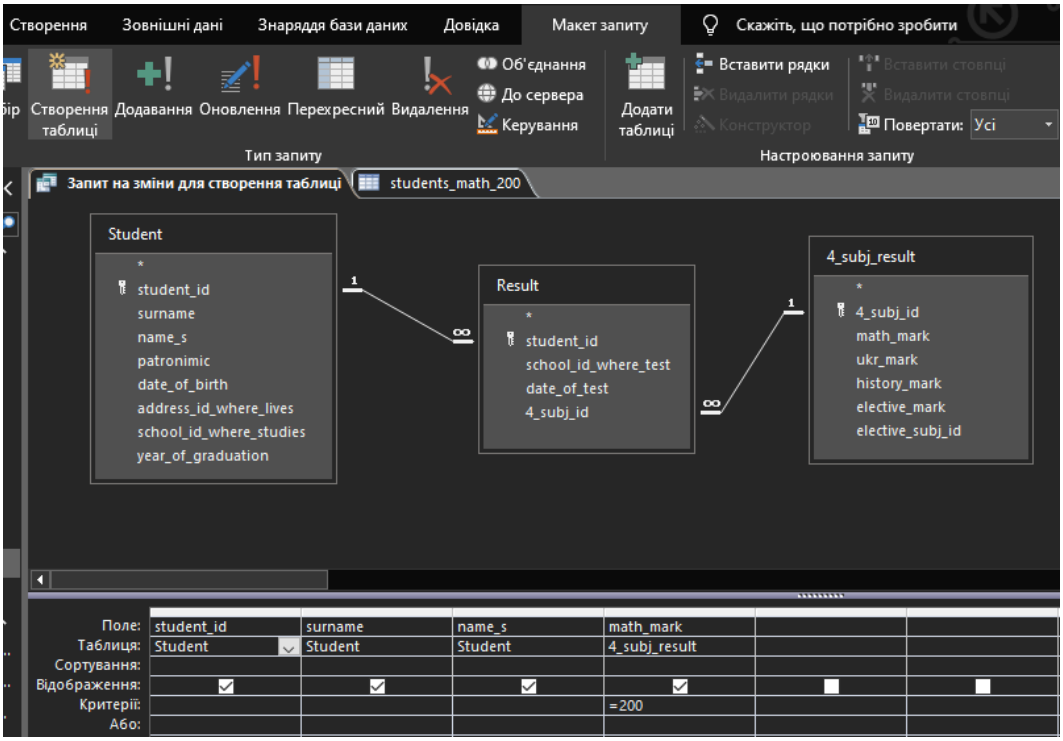
Результат:

Перехресний запит

ID школи	Школа	2023	2024	2025
1	Ліцей №1	4	3	4
2	Ліцей №2	4	3	4
3	ЗОШ №1	3	3	4
4	ЗОШ №2	3	4	3
5	Ліцей №1	3	3	2
6	Ліцей №6	3	4	1
7	ЗОШ №2	3	3	3
8	Ліцей №8	3	3	2
9	ЗОШ №2	3	3	3
10	Ліцей №10	3	3	3
11	Ліцей №11	3	4	2
12	ЗОШ №8	4	3	2
13	Ліцей №13	4	5	4
14	Ліцей №14	4	5	4
15	ЗОШ №11	4	5	4

Тип запиту: Запит на зміни для створення таблиці

Приклад: Створити таблицю учнів 200-бальників з математики.



Результат:

ID студента	Прізвище	Ім'я
15	Кіндор	Володимир
18	Козьма	Іван
27	Пастор	Андрій
28	Пономаренко	Ірина
29	Попова	Ганна
30	Романенко	Іван
31	Рудь	Ігнат
33	Севрюкова	Валентина
34	Семененко	Поліна
35	Сергієнко	Василь
36	Сулим	Володимир
37	Терлецький	Василь
38	Тимченко	Ганна
39	Турянський	Юлія
41	Шевчук	Валерій
42	Шудров	Андрій
56	Кузьміна	Ніна
57	Лементова	Катерина
61	Петренко	Михайло
62	Пилип	Катерина
63	Пономарьова	Людмила

Тип запиту: Запит на зміни для оновлення даних

Приклад: Оновити таблицю, замінивши «Чернівці» на «Івано-Франківськ».

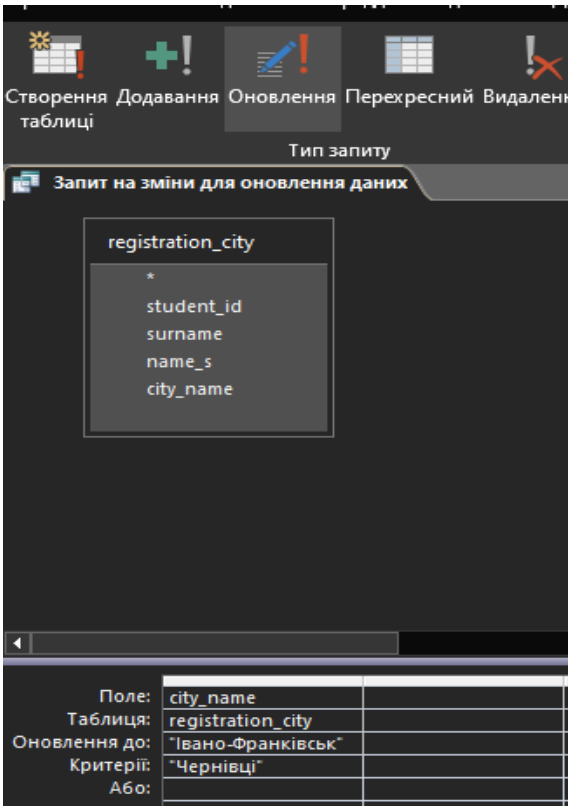
Таблиця ДО оновлення:

Створення допоміжної таблиці | registration_city

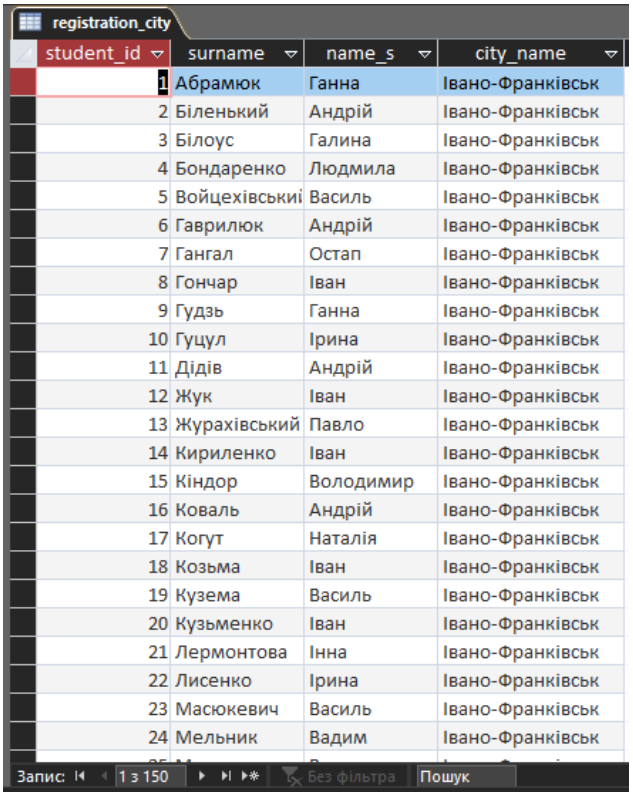
student_id	surname	name_s	city_name
1	Абрамюк	Ганна	Чернівці
2	Біленький	Андрій	Чернівці
3	Білоус	Галина	Чернівці
4	Бондаренко	Людмила	Чернівці
5	Войцехівський	Василь	Чернівці
6	Гаврилюк	Андрій	Чернівці
7	Гангал	Остап	Чернівці
8	Гончар	Іван	Чернівці
9	Гудзь	Ганна	Чернівці
10	Гуцул	Ірина	Чернівці
11	Дідів	Андрій	Чернівці
12	Жук	Іван	Чернівці
13	Журахівський	Павло	Чернівці
14	Кириленко	Іван	Чернівці
15	Кіндор	Володимир	Чернівці
16	Коваль	Андрій	Чернівці
17	Когут	Наталія	Чернівці
18	Козьма	Іван	Чернівці
19	Кузема	Василь	Чернівці
20	Кузьменко	Іван	Чернівці
21	Лермонтова	Інна	Чернівці
22	Лисенко	Ірина	Чернівці
23	Масюкевич	Василь	Чернівці
24	Мельник	Вадим	Чернівці

Запис: 1 з 150 | Без фільтра | Пошук

Запит на оновлення:

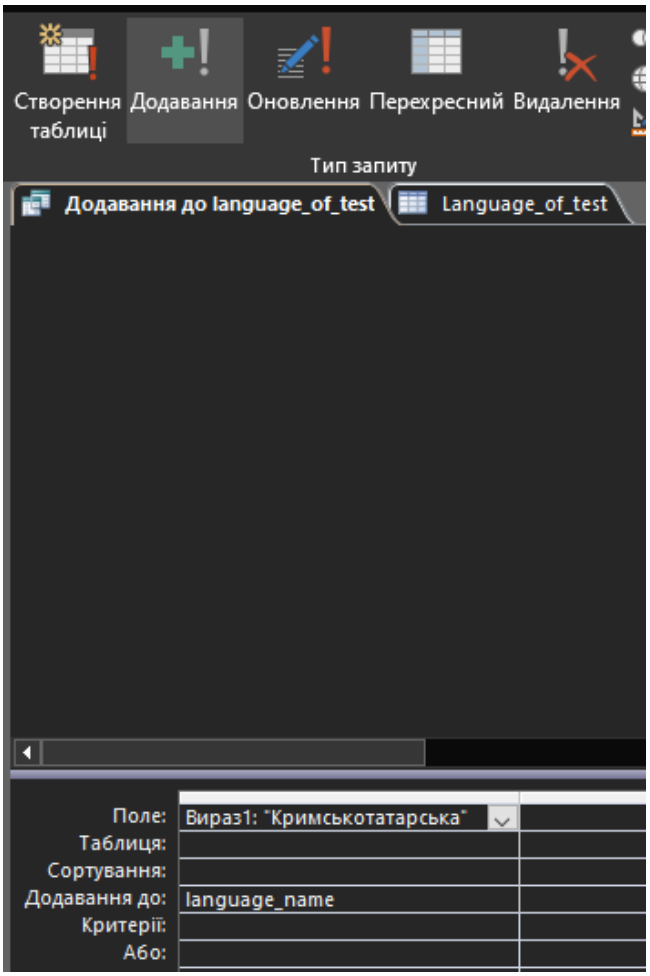


Таблиця ПІСЛЯ оновлення:

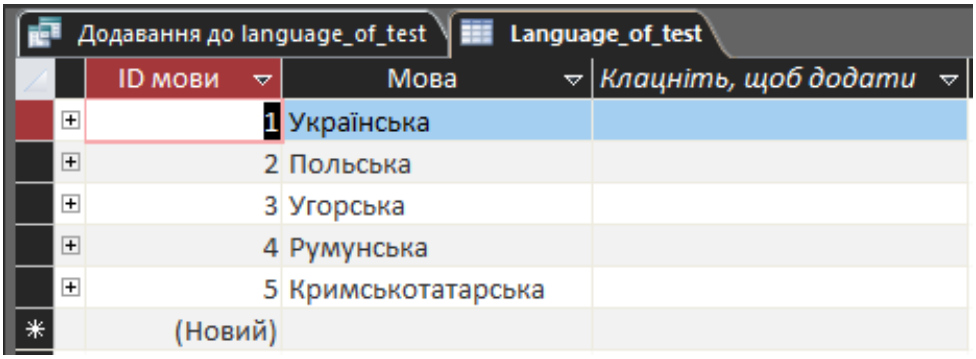


Тип запиту: Запит на зміни для добавлення даних

Приклад: Додати до таблиці мов тестування кримськотатарську.

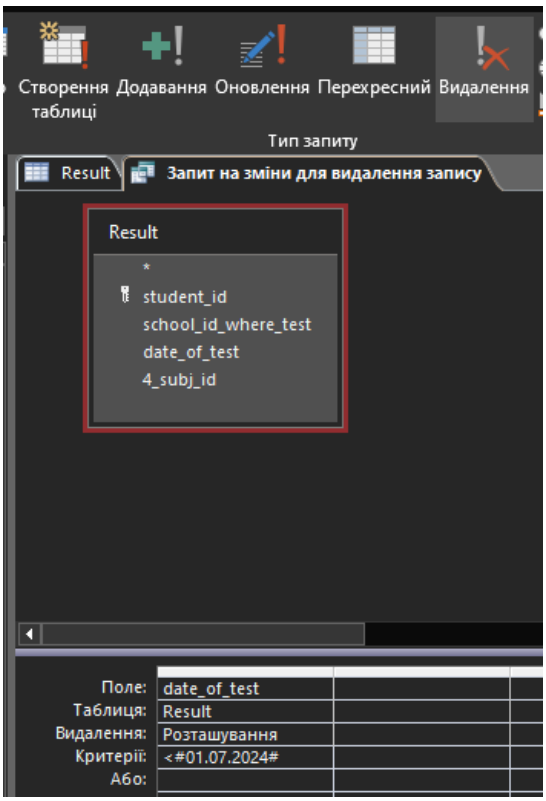


Результат:



Тип запиту: Запит на зміни для знищення даних

Приклад: Із таблиці з результатами, видалити усі записи, в яких дата складання тесту є раніше за липень 2024 року.



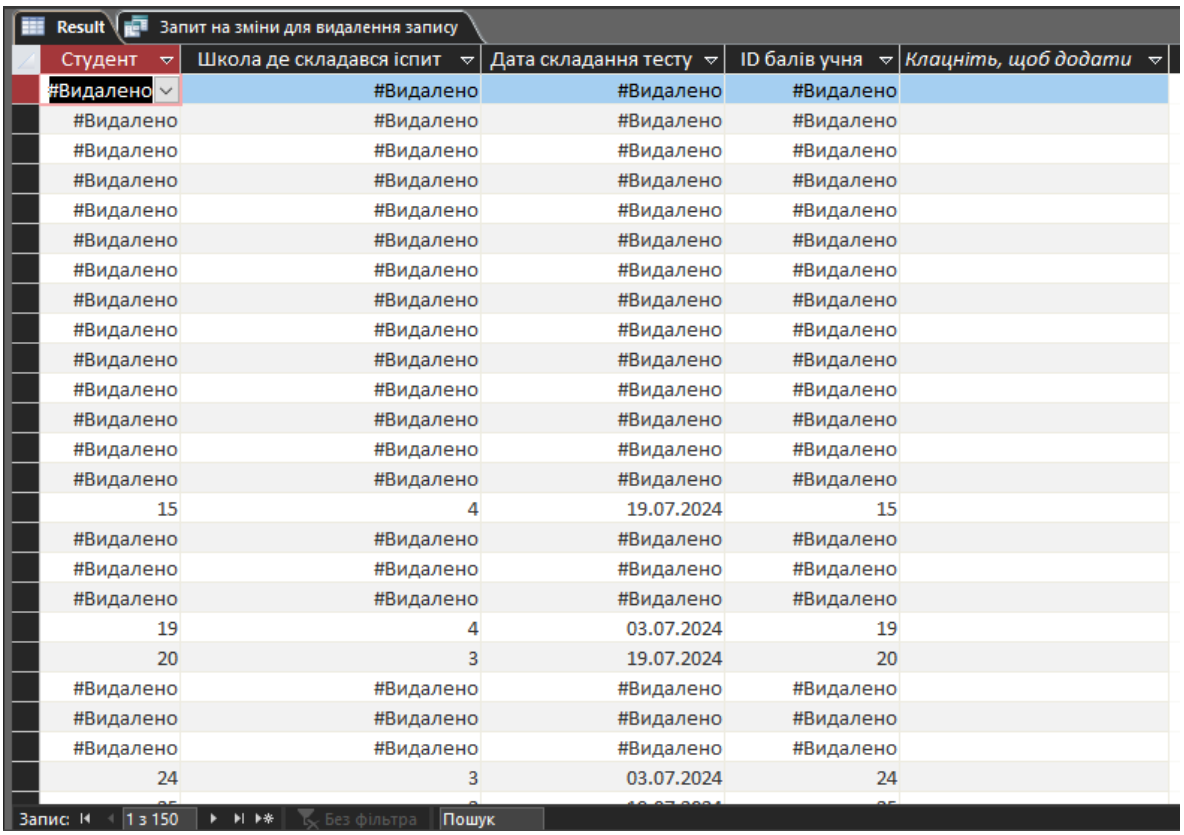
Таблиця до виконання запиту:

Result | Запит1

Студент	Школа де складався іспит	Дата складання тесту	ID балів учня	Клацніть, щоб додати
1	2	24.06.2023	1	
2	1	25.06.2023	2	
3	4	12.06.2023	3	
4	3	03.07.2023	4	
5	2	19.07.2023	5	
6	1	24.06.2023	6	
7	4	25.06.2023	7	
8	3	12.06.2023	8	
9	2	03.07.2023	9	
10	1	19.07.2023	10	
11	4	24.06.2023	11	
12	3	25.06.2023	12	
13	2	12.06.2023	13	
14	1	03.07.2023	14	
15	4	19.07.2024	15	
16	3	24.06.2024	16	
17	2	25.06.2024	17	
18	1	12.06.2024	18	
19	4	03.07.2024	19	
20	3	19.07.2024	20	
21	2	24.06.2024	21	
22	1	25.06.2024	22	
23	4	12.06.2024	23	
24	3	03.07.2024	24	

Запис: 1 з 150 Без фільтра Пошук

Після:



Result | Запит на зміни для видалення запису

Студент	Школа де складався іспит	Дата складання тесту	ID балів учня	*Клацніть, щоб додати*
#Видалено	#Видалено	#Видалено	#Видалено	
#Видалено	#Видалено	#Видалено	#Видалено	
#Видалено	#Видалено	#Видалено	#Видалено	
#Видалено	#Видалено	#Видалено	#Видалено	
#Видалено	#Видалено	#Видалено	#Видалено	
#Видалено	#Видалено	#Видалено	#Видалено	
#Видалено	#Видалено	#Видалено	#Видалено	
#Видалено	#Видалено	#Видалено	#Видалено	
#Видалено	#Видалено	#Видалено	#Видалено	
#Видалено	#Видалено	#Видалено	#Видалено	
#Видалено	#Видалено	#Видалено	#Видалено	
#Видалено	#Видалено	#Видалено	#Видалено	
#Видалено	#Видалено	#Видалено	#Видалено	
#Видалено	#Видалено	#Видалено	#Видалено	
15	4	19.07.2024	15	
#Видалено	#Видалено	#Видалено	#Видалено	
#Видалено	#Видалено	#Видалено	#Видалено	
#Видалено	#Видалено	#Видалено	#Видалено	
19	4	03.07.2024	19	
20	3	19.07.2024	20	
#Видалено	#Видалено	#Видалено	#Видалено	
#Видалено	#Видалено	#Видалено	#Видалено	
#Видалено	#Видалено	#Видалено	#Видалено	
24	3	03.07.2024	24	

Запис: 1 з 150 | Без фільтра | Пошук

Лабораторна робота №4

Створення звітів у середовищі MS Access

ЗАВДАННЯ

Використовуючи вбудовані засоби розробки звітів, побудувати звіти різної структури та складності:

1. Табличний звіт.
2. Табличний звіт з групуванням та підсумками.
3. Звіт у вільній формі.
4. Звіт у вільній формі з обчислювальними полями.
5. Звіт комбінованої структури.

МЕТОДИЧНІ ВКАЗІВКИ

1. Створити автозвіт.
2. Створити звіт за допомогою майстра звітів.
3. Створити звіт за допомогою конструктора звітів.
4. Здійснити попередній перегляд звіту.
5. Переглянути звіт.
6. Зберегти звіт.
7. Використати звіт.

Звіти призначені для оформлення потрібних даних з бази даних згідно з вимогами стандартів чи замовника, і виведення їх на папір.

Є такі засоби створення звітів:

- автозвіти;
- майстер звітів;
- конструктор звітів.

Оптимальним є комбінований спосіб: звіт створюють за допомогою майстра звіту, а пізніше допрацьовують оформлення вручну за допомогою конструктора.

Звіт складається з декількох частин (не всі вони є обов`язкові):

- заголовок звіту;
- верхній колонтитул (повторюється на всіх сторінках);
- заголовок групи (якщо записи групуються);
- область даних;
- нижній колонтитул;
- примітки.

За допомогою елементів керування у звіт можна додавати рисунки, діаграми, інші об`єкти. Дані редагувати у звіті не можна.

У формах і звітах використовується багато однакових типів елементів керування, до того ж способи їх розробки дуже подібні. Форми призначені для інтерактивного користування, а звіти використовуються для відображення або друку окремих і підсумкових значень для груп записів. Введення даних і їх редагування виконується тільки у формах. У звітах записи можна відображати або друкувати в групах, а за допомогою форм виконувати це не можна.

Створення звіту передбачає такі дії:

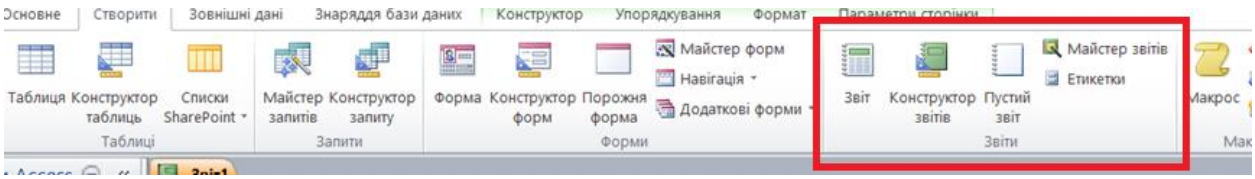
а) визначення макету звіту;

б) збір даних;

в) проектування звіту;

г) перегляд та друк звіту.

Першим етапом проектування звіту є відкриття вкладки **Створити** група **Звіти** (рис. 26).



Способи створення звітів:

- Звіт
- Конструктор звітів
- Пустий звіт
- Майстер звітів
- Етикетки

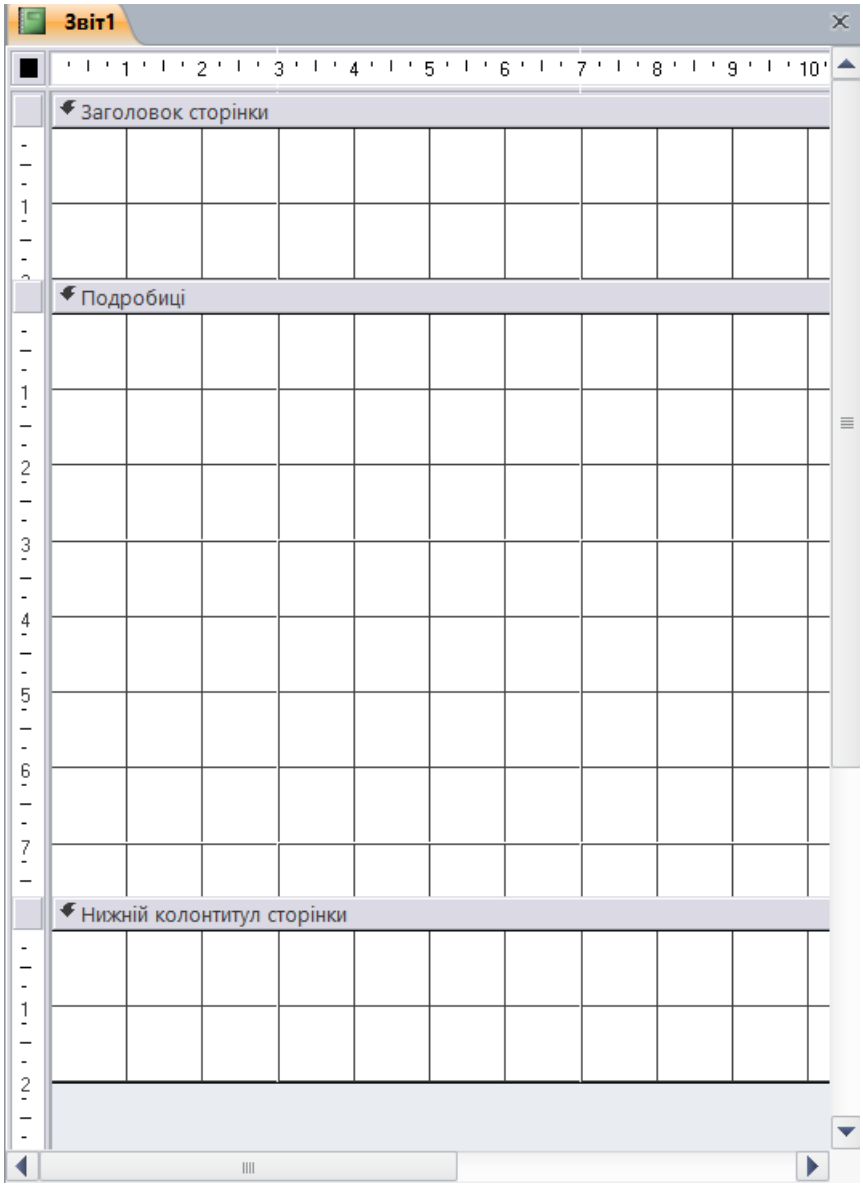
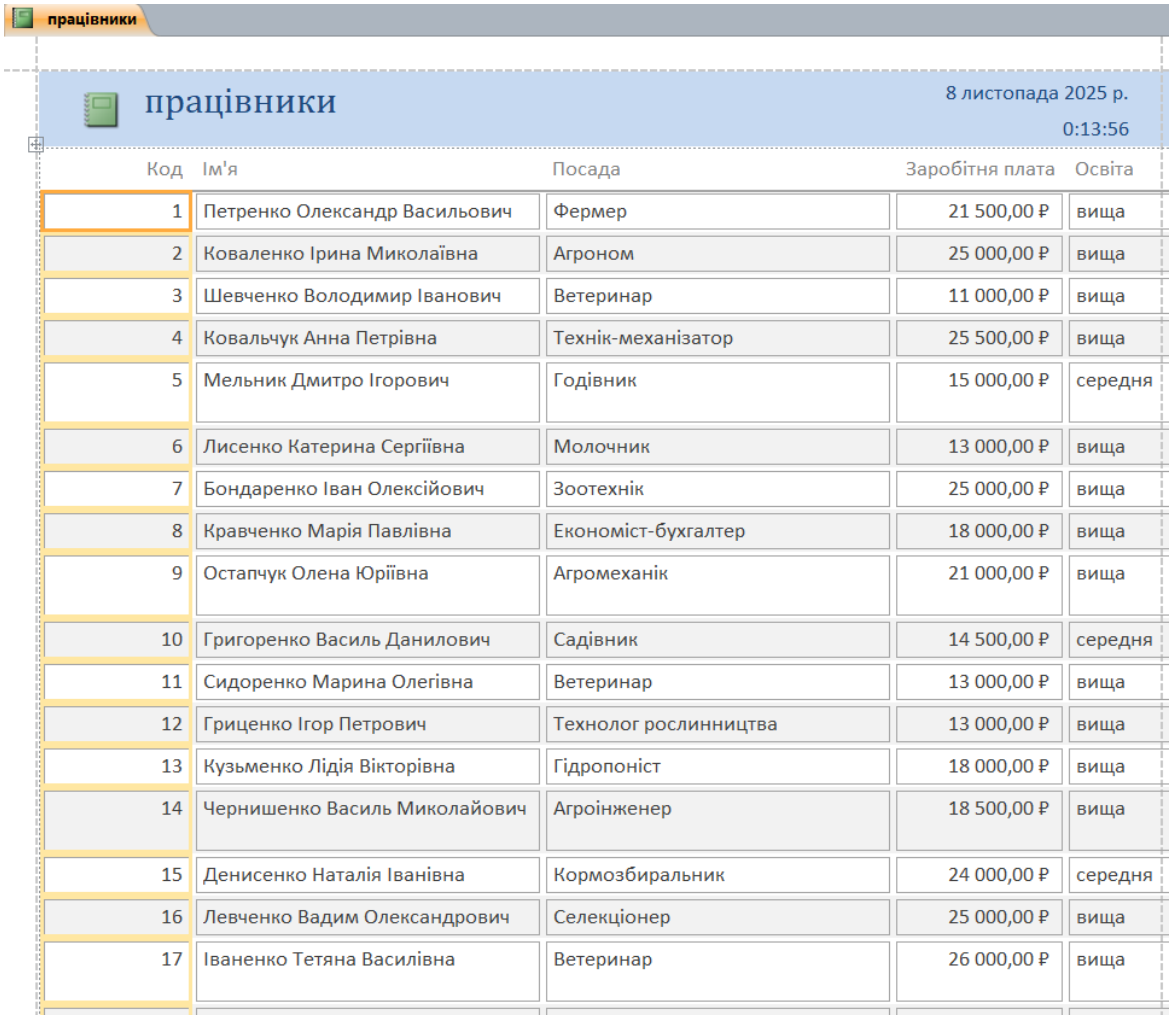


Рис. 26. Вікно порожнього звіту

1. Автоматичне створення звіту

Для автоматичного створення звіту необхідно вибрати потрібний тип **Звіт** і вказати таблицю/запит, за якою будується звіт. Після натискання кнопки **ОК**

на екран буде виведена інформація із зазначеного джерела, згідно із заданим типом звіту.



працівники

працівники

8 листопада 2025 р.
0:13:56

Код	Ім'я	Посада	Заробітня плата	Освіта
1	Петренко Олександр Васильович	Фермер	21 500,00 ₽	вища
2	Коваленко Ірина Миколаївна	Агроном	25 000,00 ₽	вища
3	Шевченко Володимир Іванович	Ветеринар	11 000,00 ₽	вища
4	Ковальчук Анна Петрівна	Технік-механізатор	25 500,00 ₽	вища
5	Мельник Дмитро Ігорович	Годівник	15 000,00 ₽	середня
6	Лисенко Катерина Сергіївна	Молочник	13 000,00 ₽	вища
7	Бондаренко Іван Олексійович	Зоотехнік	25 000,00 ₽	вища
8	Кравченко Марія Павлівна	Економіст-бухгалтер	18 000,00 ₽	вища
9	Остапчук Олена Юріївна	Агромеханік	21 000,00 ₽	вища
10	Григоренко Василь Данилович	Садівник	14 500,00 ₽	середня
11	Сидоренко Марина Олегівна	Ветеринар	13 000,00 ₽	вища
12	Гриценко Ігор Петрович	Технолог рослинництва	13 000,00 ₽	вища
13	Кузьменко Лідія Вікторівна	Гідропоніст	18 000,00 ₽	вища
14	Чернишенко Василь Миколайович	Агроінженер	18 500,00 ₽	вища
15	Денисенко Наталія Іванівна	Кормозбиральник	24 000,00 ₽	середня
16	Левченко Вадим Олександрович	Селекціонер	25 000,00 ₽	вища
17	Іваненко Тетяна Василівна	Ветеринар	26 000,00 ₽	вища

Такий спосіб створення звіту є найшвидшим, оскільки система самостійно формує макет, розташування полів і загальні елементи оформлення. Надалі користувач може за потреби перейти до режиму конструктора або режиму макета, щоб відкоригувати зовнішній вигляд, додати обчислювальні поля, заголовки, групування чи інші елементи звіту.

2. Створення звіту за допомогою майстра звіту

Обравши тип створення **Майстер звітів** відкривається діалогове вікно

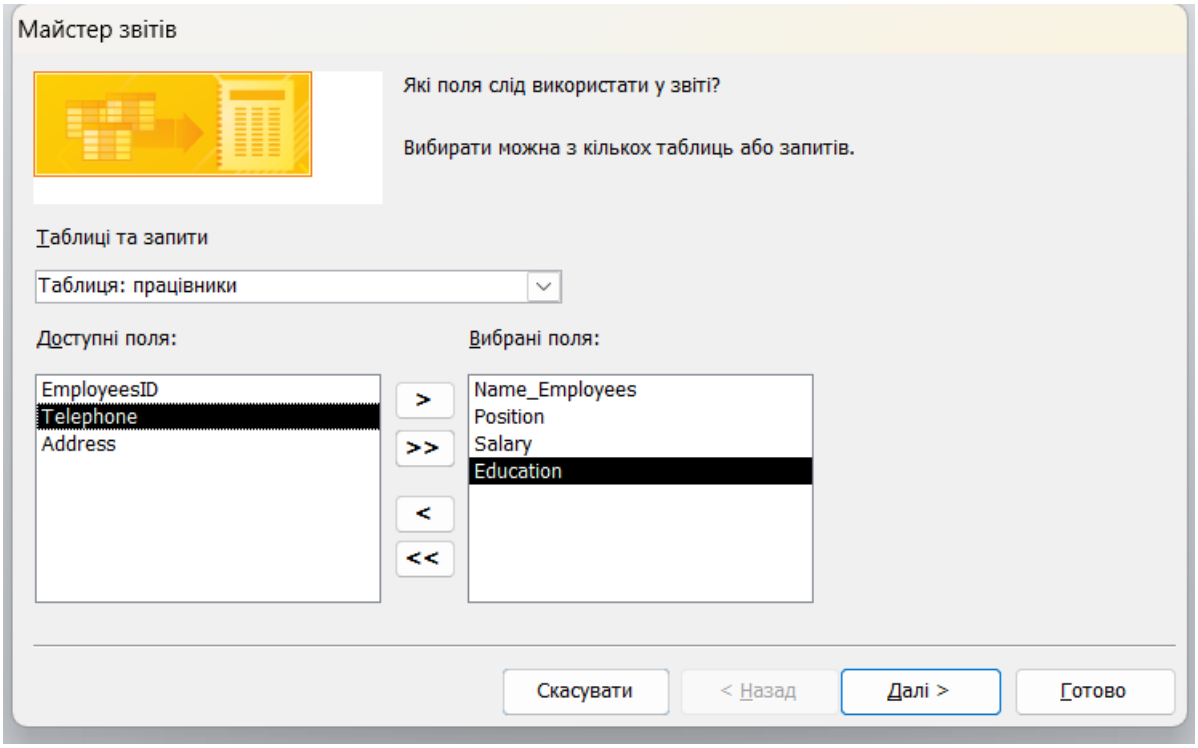


Рис.27. Перше діалогове вікно **Майстер звітів**

У діалоговому вікні (рис.27) необхідно вибрати джерело даних (таблиця чи запит)

У цьому ж вікні відбираємо потрібні поля із списку **Доступні поля** і переносимо їх у список **Вибрані поля**, натискаємо кнопку **Далі** і переходимо в друге вікно майстра звіту (рис. 28).

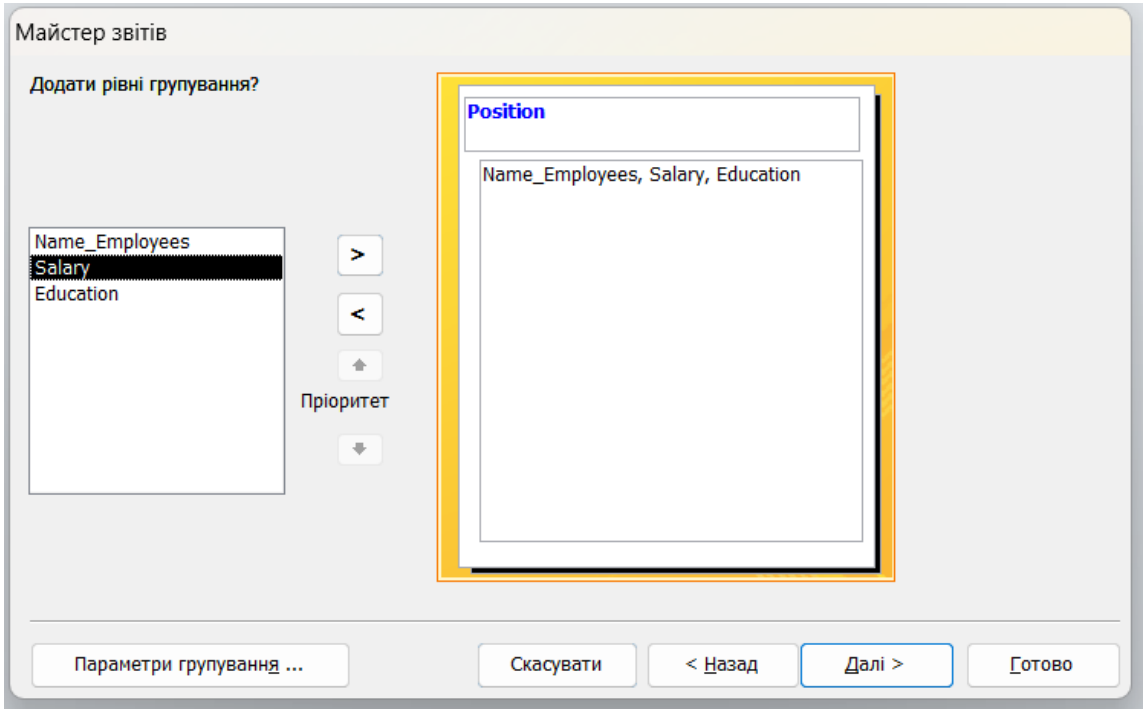


Рис. 28. Друге діалогове вікно **Майстер звітів**

Якщо потрібно визначити поле (або поля), яке буде використане для групування записів необхідно вибрати його (їх) і клацнути по кнопці зі стрілкою вправо, щоб добавити розділ групи. Задавши групування, клацнути на кнопці **Далі**.

Переходимо в наступне діалогове вікно сортування записів, які будуть виводитись у звіт (рис. 29).

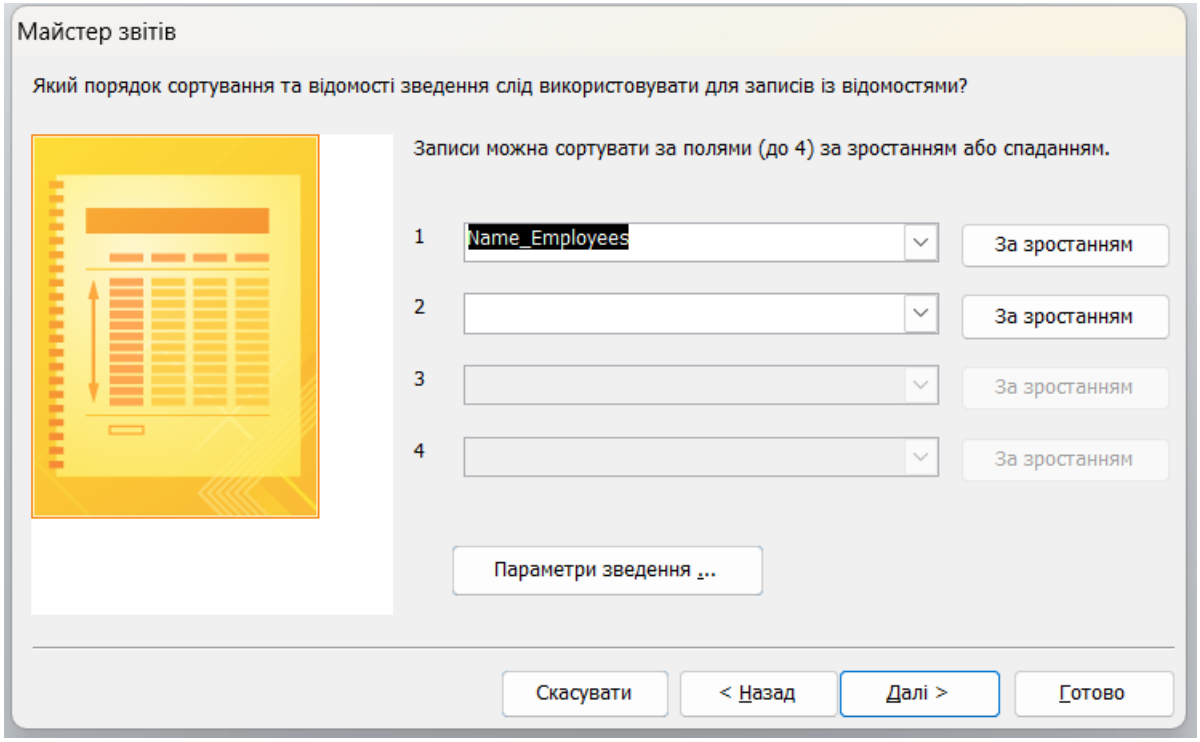


Рис.29. Третє діалогове вікно **Майстер звітів**

За замовчуванням записи сортуються в зростаючому порядку за полем, по якому є групування, а далі послідовно за полями, які виводяться у звіт.

Якщо порядок сортування не змінюється, то у діалоговому вікні натискаємо кнопку **Далі** і переходимо у наступне вікно майстра звіту.

У наступному діалоговому вікні, яке з`явилося, необхідно вказати макет (Східчастий, блок чи структура), орієнтацію (книжкова або альбомна) і необхідність автоматичного налаштування ширини полів, щоб помістити їх на одну сторінку.

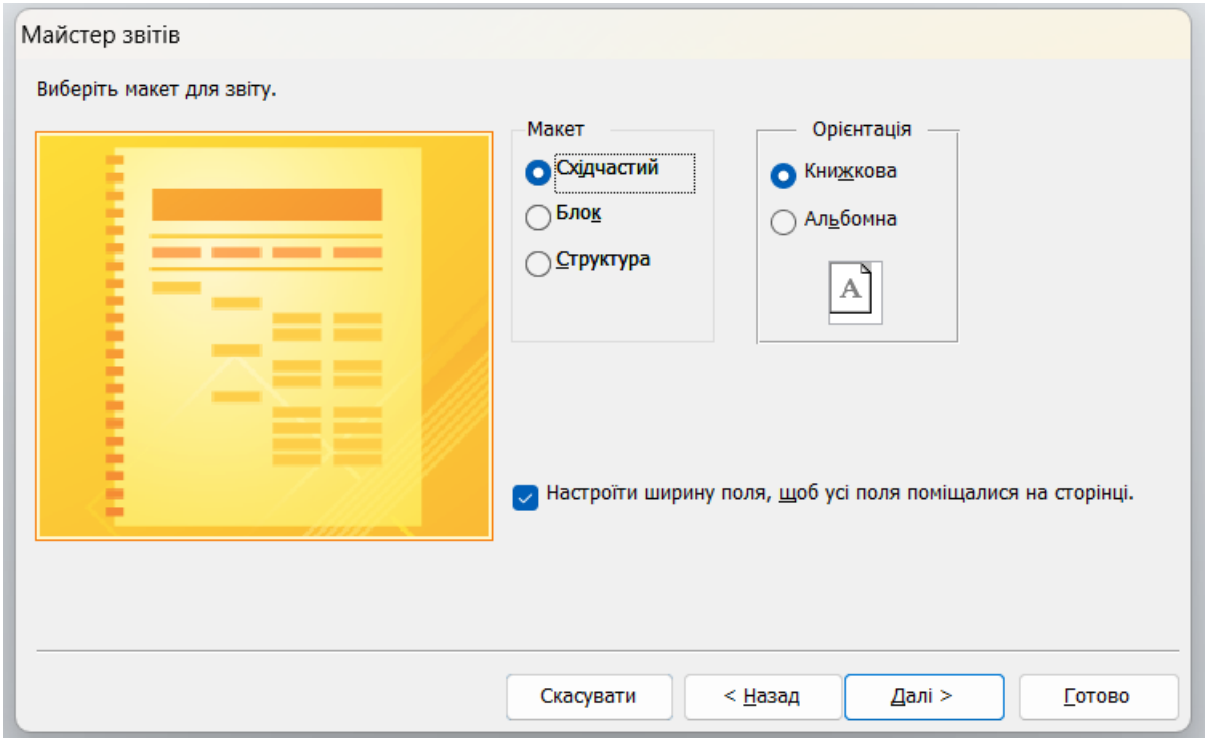


Рис.30. Четверте діалогове вікно **Майстер звітів**

Виконавши необхідні налаштування, натискуємо кнопку **Далі.** Переходимо в наступне діалогове вікно майстра звітів.

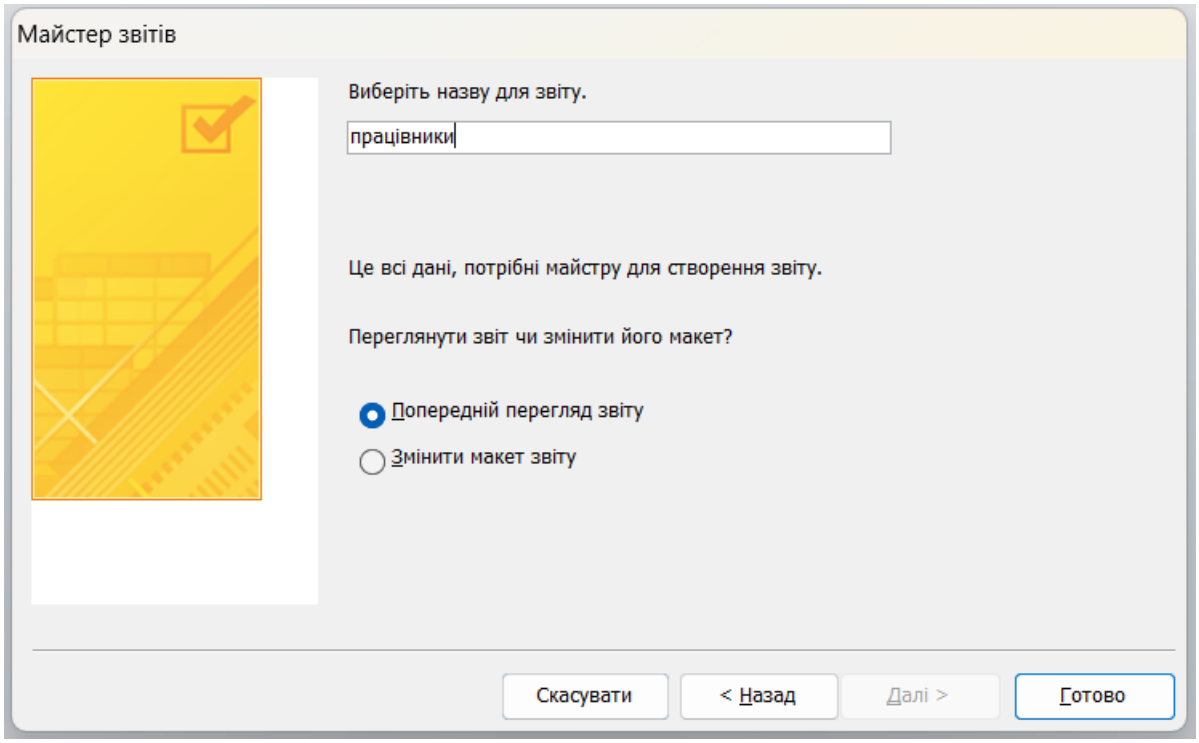


Рис. 31. П’яте діалогове вікно **Майстер звітів**

Задаємо заголовок звіту і натискуємо кнопку **Готово**. На екрані з’являється створений звіт.

Працівники

Посада	Ім'я	Заробітня плата	Освіта
Агроеколог			
	Гриценко Олександр Ігорович	24 500,00 ₴	вища
	Гриценко Тамара Петрівна	20 000,00 ₴	вища
	Кравченко Владислава Миколаївна	15 500,00 ₴	вища
	Мельниченко Тетяна Павлівна	11 000,00 ₴	вища
	Романчук Олександра Ігорівна	25 000,00 ₴	вища
Агроінженер			
	Ковальчук Вікторія Анатоліївна	18 500,00 ₴	вища
	Литвиненко Олексій Ігорович	21 500,00 ₴	вища
	Литвиненко Олена Андріївна	15 500,00 ₴	вища
	Литвинова Ірина Петрівна	11 000,00 ₴	вища
	Павлюк Володимир Іванович	22 000,00 ₴	вища
	Чернишенко Василь Миколайович	18 500,00 ₴	вища
Агроменеджер			
	Ігнатенко Вікторія Петрівна	13 000,00 ₴	вища
	Ковальчук Іван Іванович	11 500,00 ₴	вища
	Медведенко Ірина Олександрівна	24 500,00 ₴	вища
	Черненко Ірина Петрівна	12 000,00 ₴	вища
Агромеханік			
	Волошин Олег Валентинович	28 000,00 ₴	вища
	Остапчук Олена Юріївна	21 000,00 ₴	вища

3. Створення звіту вручну за допомогою конструктора

Якщо звіти, які створюються за замовчуванням або за допомогою майстра звітів, не задовольняють вимоги, необхідно розробляти їх вручну. Можна відкрити порожній звіт і добавити в нього поля, верхній і нижній колонтитули, визначити спосіб групування, сортування. Для створення такого звіту необхідно у вікні бланка бази даних (рис. 3) вибрати вкладинку **Створити** групи **Звіти.** Обираємо пункт **Конструктор звітів**, в результаті у вікні конструктора з’явиться порожній звіт (рис.32).

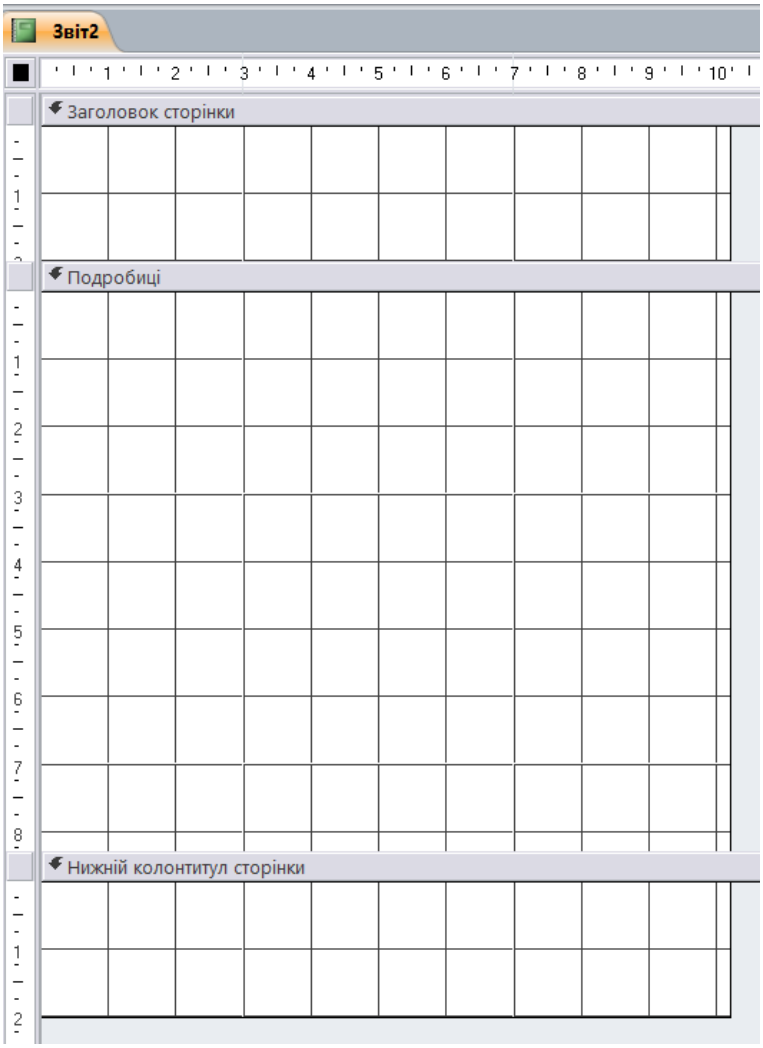


Рис.32. Вікно порожнього звіту

За замовчуванням кожний звіт, який створюється містить такі розділи: *заголовок звіту, верхній колонтитул, область даних, нижній колонтитул, примітки звіту*. Якщо при створенні звіту здійснюється групування записів, то добавляється новий розділ *заголовка групи*.

Розділ *верхнього колонтитула* з`являється вверху кожної сторінки. Він може містити заголовки стовпців табличного звіту. Відповідно розділ *нижнього колонтитула* з`являється внизу кожної сторінки і, як правило, містить номери сторінок.

Розділ *заголовка звіту* є необов`язковим розділом, який містить інформацію, що з`являється тільки один раз – на початку звіту.

Розділ *приміток звіту* аналогічний розділу *заголовка*, але друкується один раз в кінці звіту. Як правило, він використовується для розміщення підсумкових значень.

Для того, щоб добавити розділи *заголовка* і *приміток* необхідно викликати контекстне меню команду **Заголовок/Примітка звіту.**

Розділ *область даних* містить основну частину даних, які необхідно подати у звіті. Тут поміщається інформація із полів бази даних або результат обчислень над полями. Елементи керування, які знаходяться в цьому розділі, з`являються для кожного запису, вибраного із бази даних.

Розділ *заголовка/примітки групи* – необов`язковий розділ, який з`являється на початку/в кінці кожної групи записів. При добавленні *заголовка групи* вказується спосіб групування даних (поле) у звіті по цьому заголовку. Дані звіту будуть відсортовані по цьому полю і згруповані так, що для кожної групи відповідна назва поля буде виводиться один раз. У розділі *примітки групи* поміщаються підсумки (розрахунки) по групах.

Заповнення цих розділів здійснюється як і при заповненні відповідних розділів при створенні форм за допомогою конструктора (див. завдання 3 до лабораторної роботи №4).

Часто потрібно, щоб дані, які відображаються у звіті, були подані в певному порядку (відсортовані). Крім того, у звіті дані можна групувати за категоріями, а в категоріях – за підкатегоріями. Сортування й групування зв`язані між собою, оскільки сортування даних приводить до їх групування.

Для сортування й групування використовуються параметри діалогового вікна "**Сортування і групування**" (рис.33). Щоб відкрити це вікно, необхідно виконати команду **Сортування і групування** із контекстного меню.

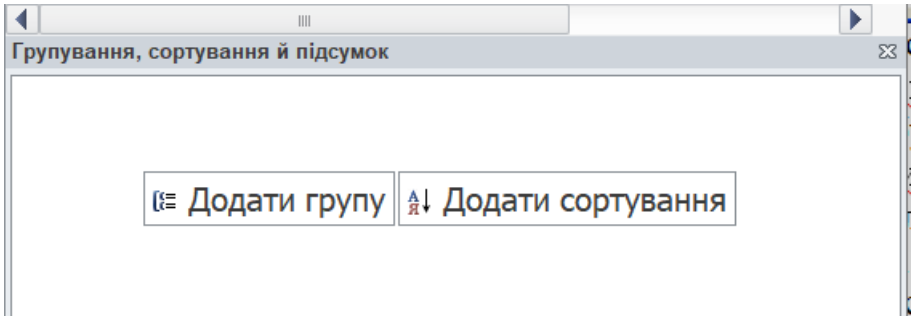


Рис. 33 Діалогове вікно " **Сортування і групування** "

У першому стовпці **Додати групу** вікна розкриваємо список і вибираємо ім`я поля, за яким буде здійснюватися групування. В другому ствопці **Додати**

сортування задаємо спосіб сортування за полем групування. Далі встановлюємо властивості групи: активуємо *Заголовок групи* і *Примітка групи.* Ці дії виконуємо для всіх визначених рівнів сортування й групування (в Access допускається максимум 10 рівнів). Пріоритет сортування й групування визначається в діалоговому вікні зверху вниз. Отже, перший рядок у стовпці **Поле** визначає найвищий рівень сортування й групування, а останній – найнижчий. У звіт, як і у форми можна добавляти обчислювальні елементи керування для відображення даних. В обчисленнях можна використати будь-які допустимі функціїї Access (наприклад Date()– поточна дата, Avg(значення) – середнє значення, Max(значення) – максимальне значеня і т. д.). У звіті можна відобразити номер сторінки (Page) і загальну кількість сторінок (Pages).

На (рис. 34) зображено макет звіту, який містить всі описані вище розділи.

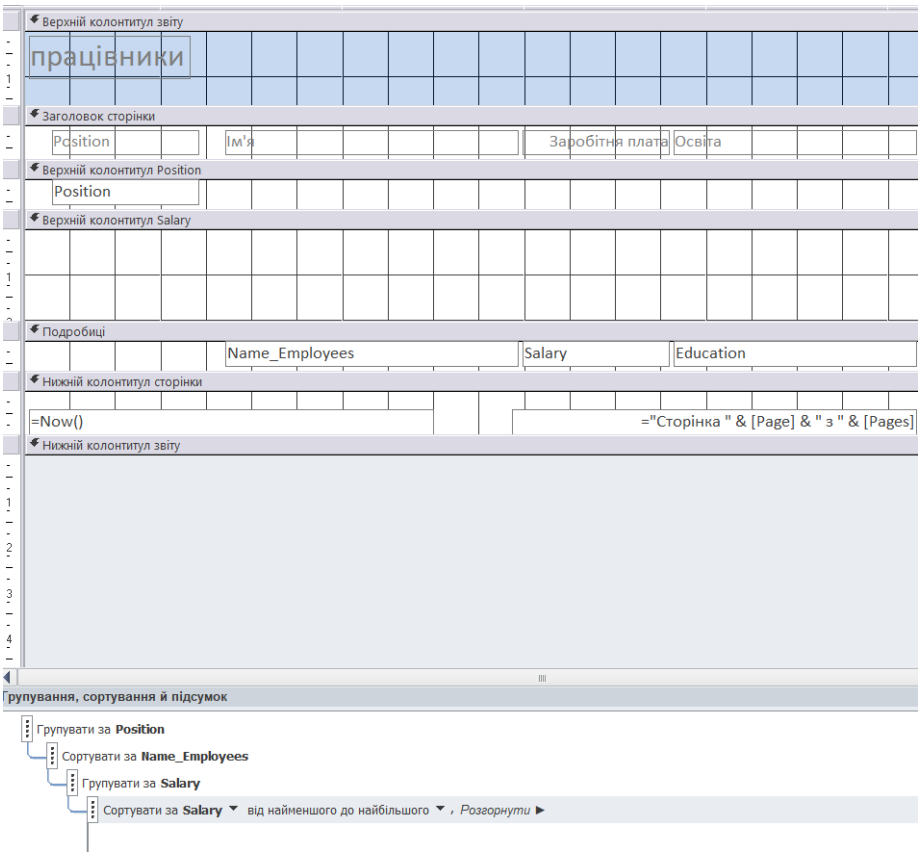
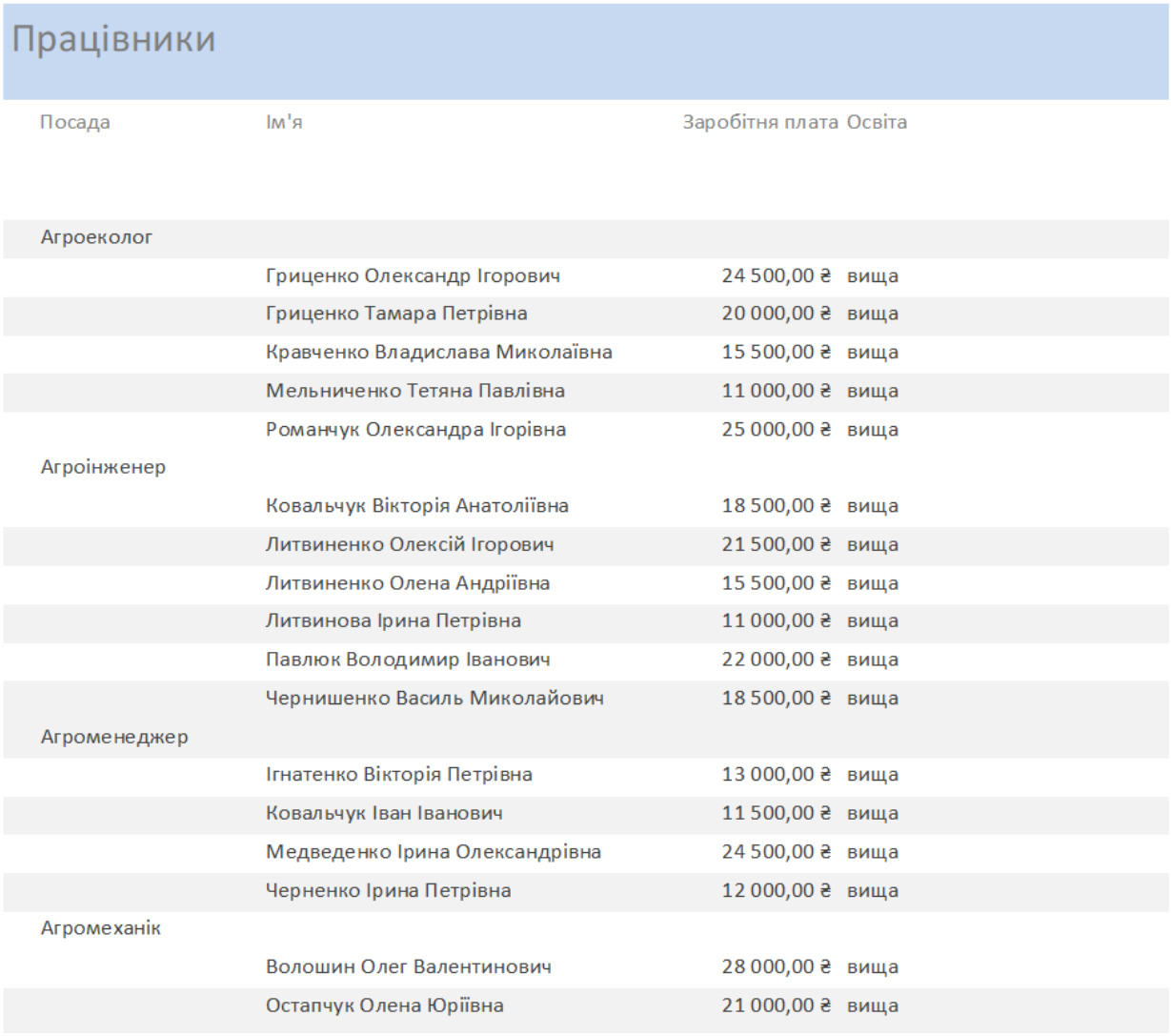


Рис. 34. Макет звіту

4. Попередній перегляд звіту

На будь-якій стадії розробки звіту користувач має можливість переглянути його вміст, що дозволяє оцінити зовнішній вигляд та структуру звіту до його остаточного завершення.

Для цього необхідно на панелі інструментів обрати режим подання **Вигляд**, після чого на екрані відобразиться відповідне подання звіту, що надає змогу детально ознайомитися з його оформленням і наповненням.



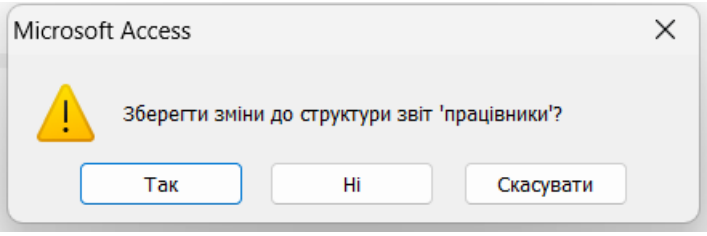
Працівники

Посада	Ім'я	Заробітня плата	Освіта
Агроеколог			
	Гриценко Олександр Ігорович	24 500,00 ₴	вища
	Гриценко Тамара Петрівна	20 000,00 ₴	вища
	Кравченко Владислава Миколаївна	15 500,00 ₴	вища
	Мельниченко Тетяна Павлівна	11 000,00 ₴	вища
	Романчук Олександра Ігорівна	25 000,00 ₴	вища
Агроінженер			
	Ковальчук Вікторія Анатоліївна	18 500,00 ₴	вища
	Литвиненко Олексій Ігорович	21 500,00 ₴	вища
	Литвиненко Олена Андріївна	15 500,00 ₴	вища
	Литвинова Ірина Петрівна	11 000,00 ₴	вища
	Павлюк Володимир Іванович	22 000,00 ₴	вища
	Чернишенко Василь Миколайович	18 500,00 ₴	вища
Агроменеджер			
	Ігнатенко Вікторія Петрівна	13 000,00 ₴	вища
	Ковальчук Іван Іванович	11 500,00 ₴	вища
	Медведенко Ірина Олександрівна	24 500,00 ₴	вища
	Черненко Ірина Петрівна	12 000,00 ₴	вища
Агромеханік			
	Волошин Олег Валентинович	28 000,00 ₴	вища
	Остапчук Олена Юріївна	21 000,00 ₴	вища

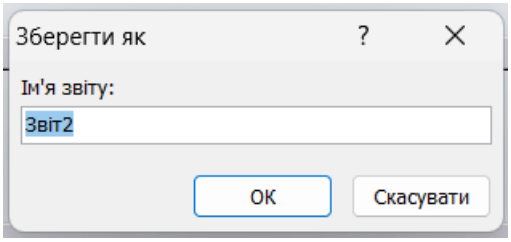
Рис.35. Вікно попереднього перегляду звіту.

5. Збереження звіту

Для збереження звіту необхідно виконати команду **Зберегти** із меню **Файл** або закрити вікно звіту і в діалоговому вікні, що з'являється, клацнути на кнопці **Так.**



Якщо звіт зберігається вперше, то відобразиться діалогове вікно, в якому потрібно задати ім`я звіту.



6.Використання звіту

Щоб відкрити збережений звіт, необхідно двічі клацнути на його піктограмі у вікні бланка бази даних або вибрати його і клацнути на кнопці **Відкрити**.

Розглянемо приклади звітів різної складності, згідно завдань лабораторної роботи.

1. **Табличний звіт**

Список учнів : ПІБ, датою народження та роком випуску зі школи.

Student

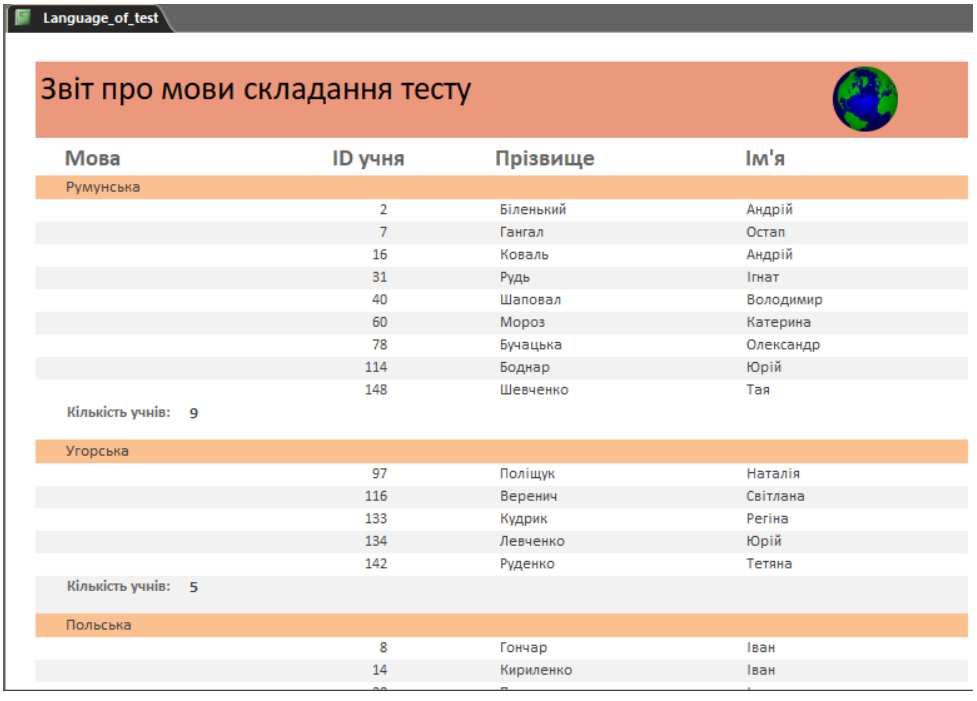
Дані про випускників

1 листопада 2025 р.

ID студента	Прізвище	Ім'я	По батькові	Дата народження	Рік випуску
1	Абрамюк	Ганна	Юріївна	07.06.2006	2023
2	Біленький	Андрій	Олександрович	14.05.2006	2023
3	Білоус	Галина	Юріївна	21.05.2006	2023
4	Бондаренко	Людмила	Олександрівна	19.02.2006	2023
5	Войцехівський	Василь	Олександрович	31.12.2006	2023
6	Гаврилюк	Андрій	Петрович	02.11.2006	2023
7	Гангал	Остап	Васильович	02.04.2006	2023
8	Гончар	Іван	Петрович	25.06.2006	2023
9	Гудзь	Ганна	Олегівна	19.05.2006	2023
10	Гуцул	Ірина	Михайлівна	03.10.2006	2023
11	Дідів	Андрій	Юрійович	01.11.2006	2023
12	Жук	Іван	Іванович	09.01.2006	2023
13	Журахівський	Павло	Олександрович	25.06.2006	2023
14	Кириленко	Іван	Андрійович	20.04.2006	2023
15	Кіндор	Володимир	Петрович	17.07.2007	2024
16	Коваль	Андрій	Юрійович	21.02.2007	2024
17	Когут	Наталія	Володимирівна	04.01.2007	2024
18	Козьма	Іван	Сергійович	30.11.2007	2024
19	Кузема	Василь	Сергійович	26.04.2007	2024
20	Кузьменко	Іван	Васильович	29.03.2007	2024
21	Лермонтова	Інна	Петрівна	15.03.2007	2024
22	Лисенко	Ірина	Михайлівна	05.01.2007	2024

2. Табличний звіт з групуванням та підсумками

Звіт, згрупований за мовою складання тесту та з підсумками про кількість учнів, що складала кожною мовою.



Language_of_test

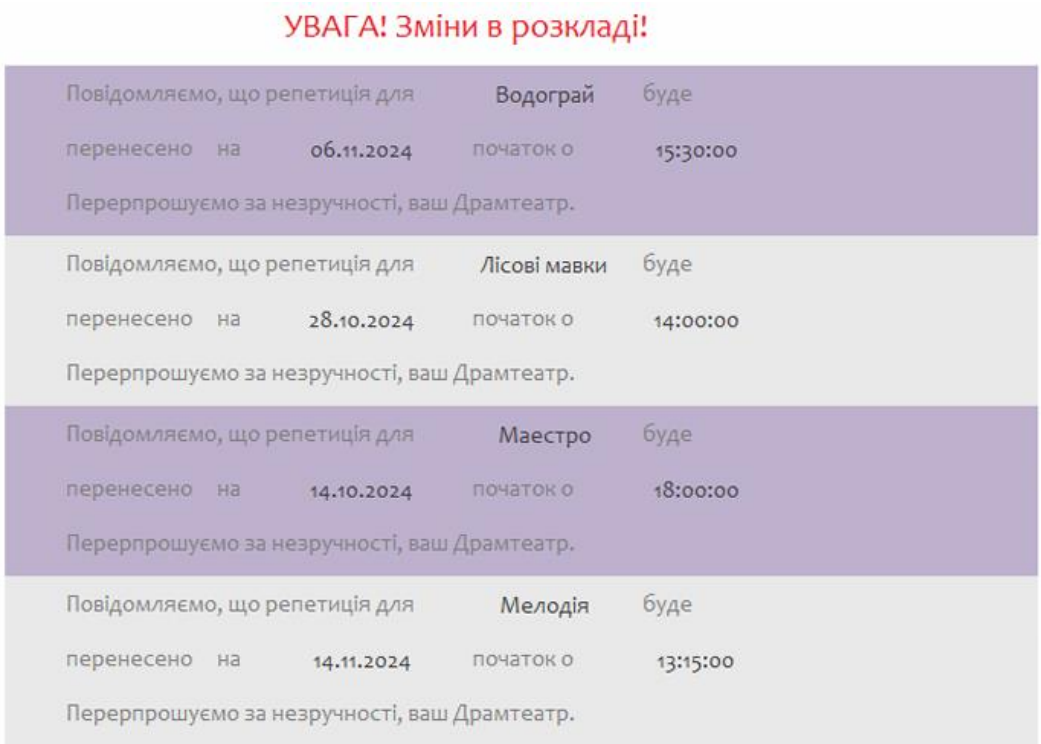
Звіт про мови складання тесту

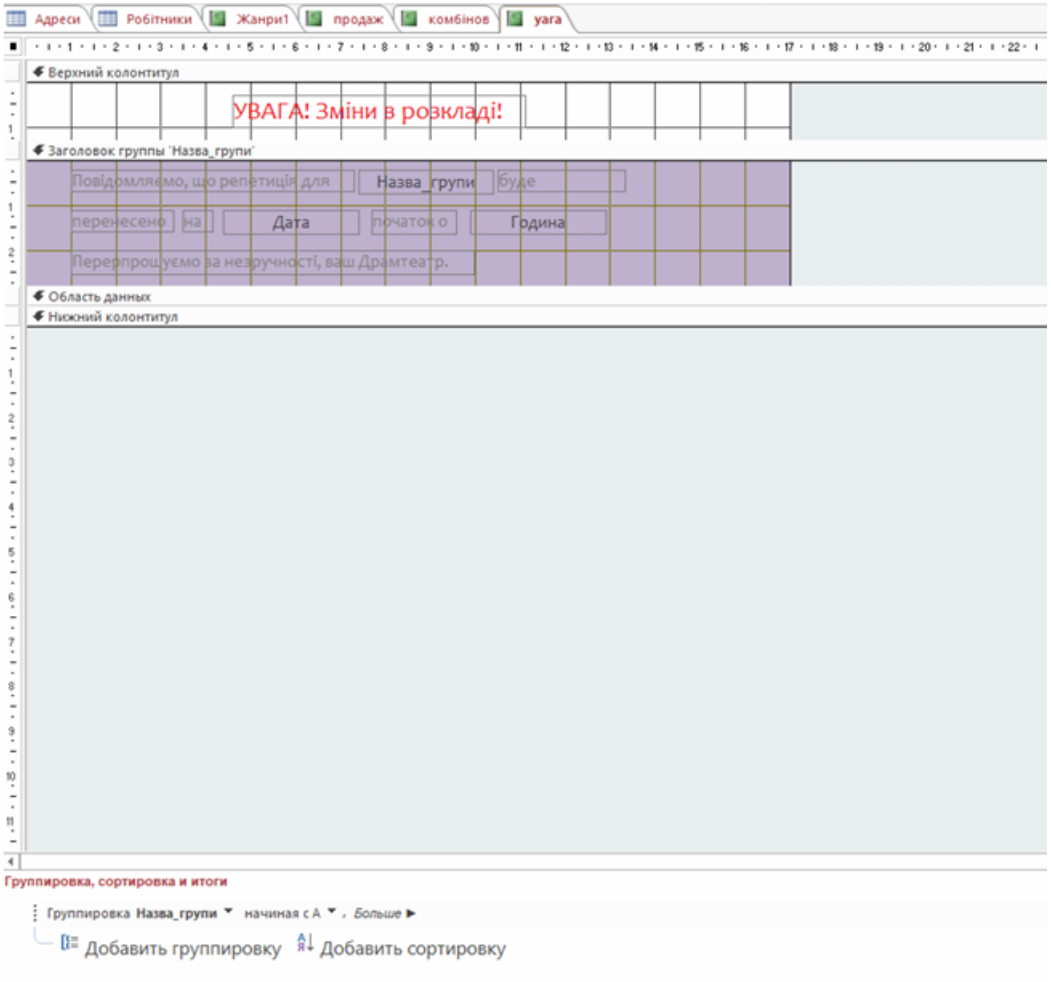
Мова	ID учня	Прізвище	Ім'я
Румунська			
	2	Біленький	Андрій
	7	Гангал	Остап
	16	Коваль	Андрій
	31	Рудь	Ігнат
	40	Шаповал	Володимир
	60	Мороз	Катерина
	78	Бучацька	Олександр
	114	Боднар	Юрій
	148	Шевченко	Тая
Кількість учнів: 9			
Угорська			
	97	Поліщук	Наталія
	116	Верених	Світлана
	133	Кудрик	Регіна
	134	Левченко	Юрій
	142	Руденко	Тетяна
Кількість учнів: 5			
Польська			
	8	Гончар	Іван
	14	Кириленко	Іван

3. Звіт у вільній формі

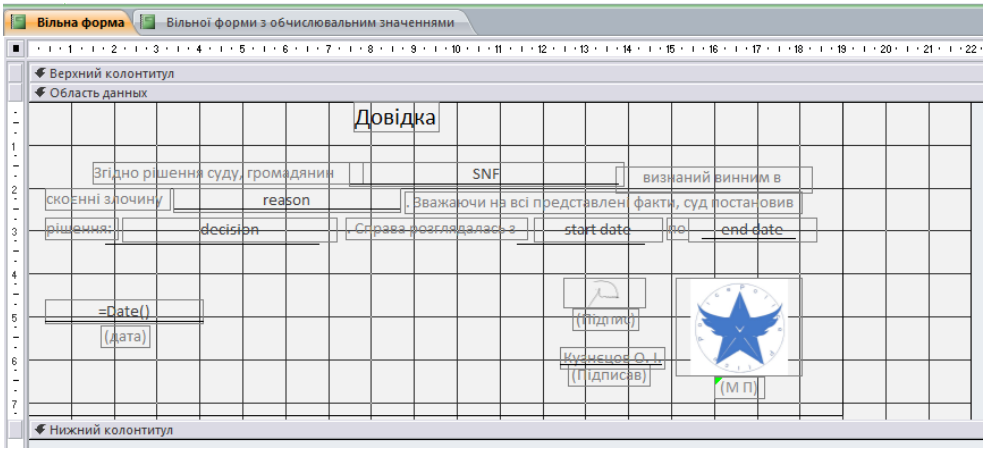
Приклад1. Етикетки з реєстраційними даними кожного учня.

Приклад2. Інформація про переніс репетиції для кожної групи з датою та часом.



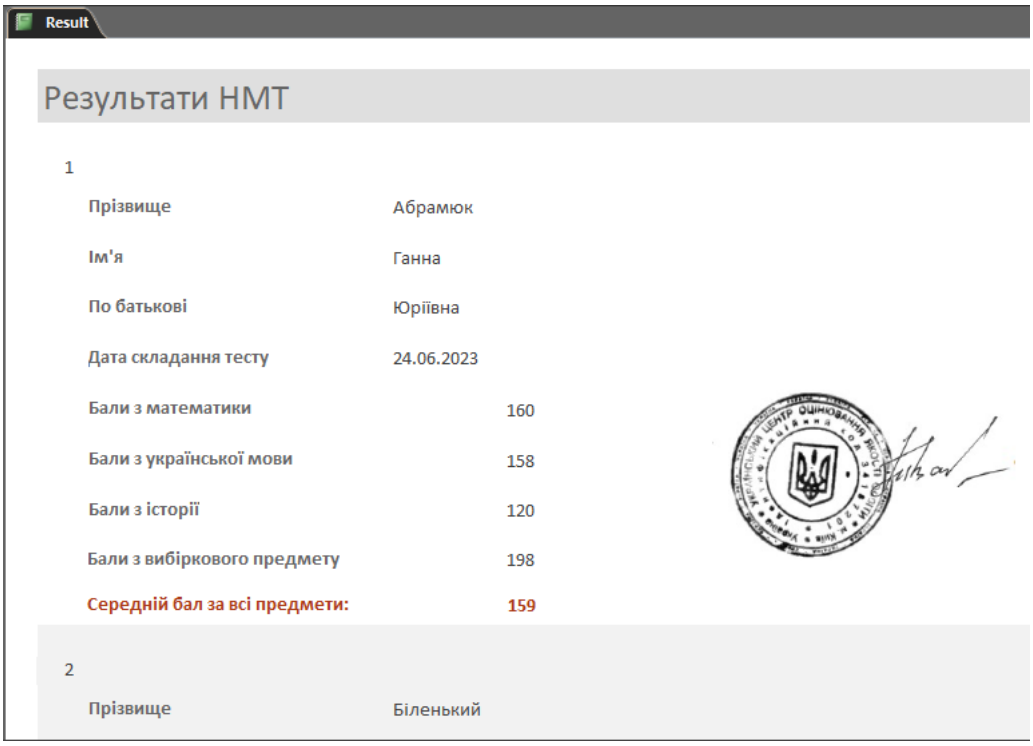


Приклад3. Зробити довідки на кожного громадянина, де справа вже є розглянутою.

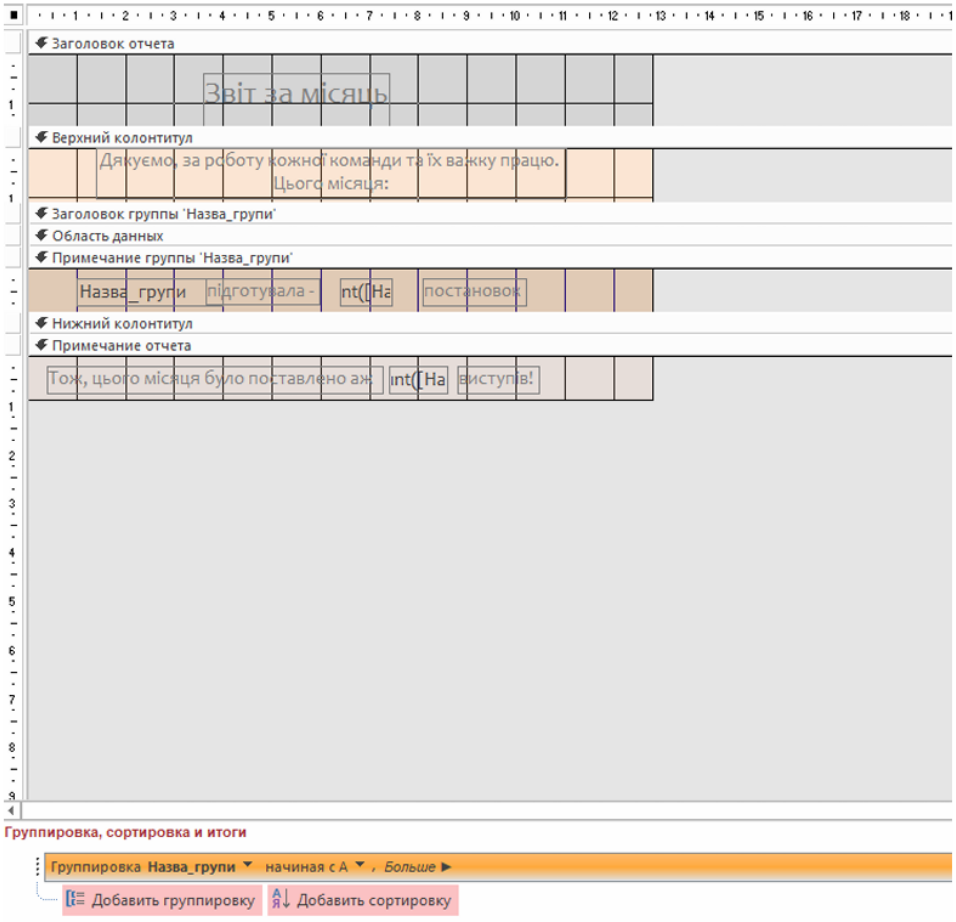


4. Звіт у вільній формі з обчислювальними полями

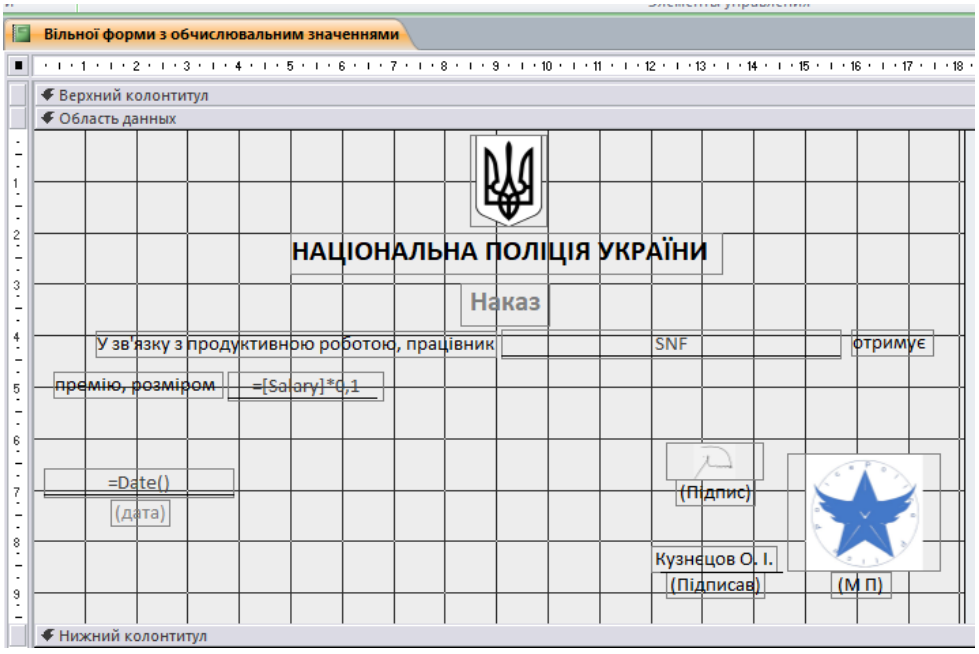
Приклад1. Звіт про бали кожного учня разом з обчисленням середньої оцінки за всі предмети.



Приклад2. Звіт про кількість постановок кожної групи цього місяця та загальна їх сума.



Приклад3. Видати наказ на премію для кожного працівника



5. Звіт комбінованої структури

Приклад1. Звіт з групуванням за школою, яку закінчив учень, та містом, у якому вона знаходиться, з відображенням тих випускників, що набрали 170+ балів з математики.

School

Випускники, що мають 170+ балів з математики

Школа	Місто	Прізвище	Ім'я	Бали з математики
ЗОШ №1	Чернівці			
		Пастор	Андрій	171
		Кіндор	Володимир	170
ЗОШ №11	Тернопіль			
		Данилюк	Тетяна	200
		Яценко	Сергій	188
		Гордашко	Роман	176
		Кравчук	Олена	189
		Лисий	Арсен	194
		Хоменко	Юрій	190
ЗОШ №2	Чернівці			
		Жук	Іван	176
		Пономаренко	Ірина	179
		Мельник	Вадим	193
		Кузьменко	Іван	173
		Гавалешко	Надія	186
		Даниленко	Любов	191
		Семенчук	Людмила	183

Приклад2. Вивести інформацію про адвокатів і їхню спеціалізацію

Довідка

Іванов Петро Сергійович

Телефон: +380501234567

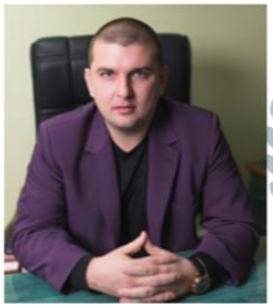
Стаж роботи: 19 р.

Даний адвокат, спеціалізується над такими роботами:

1	Порушення ПДР

Довідка

Петров Іван Олексійович

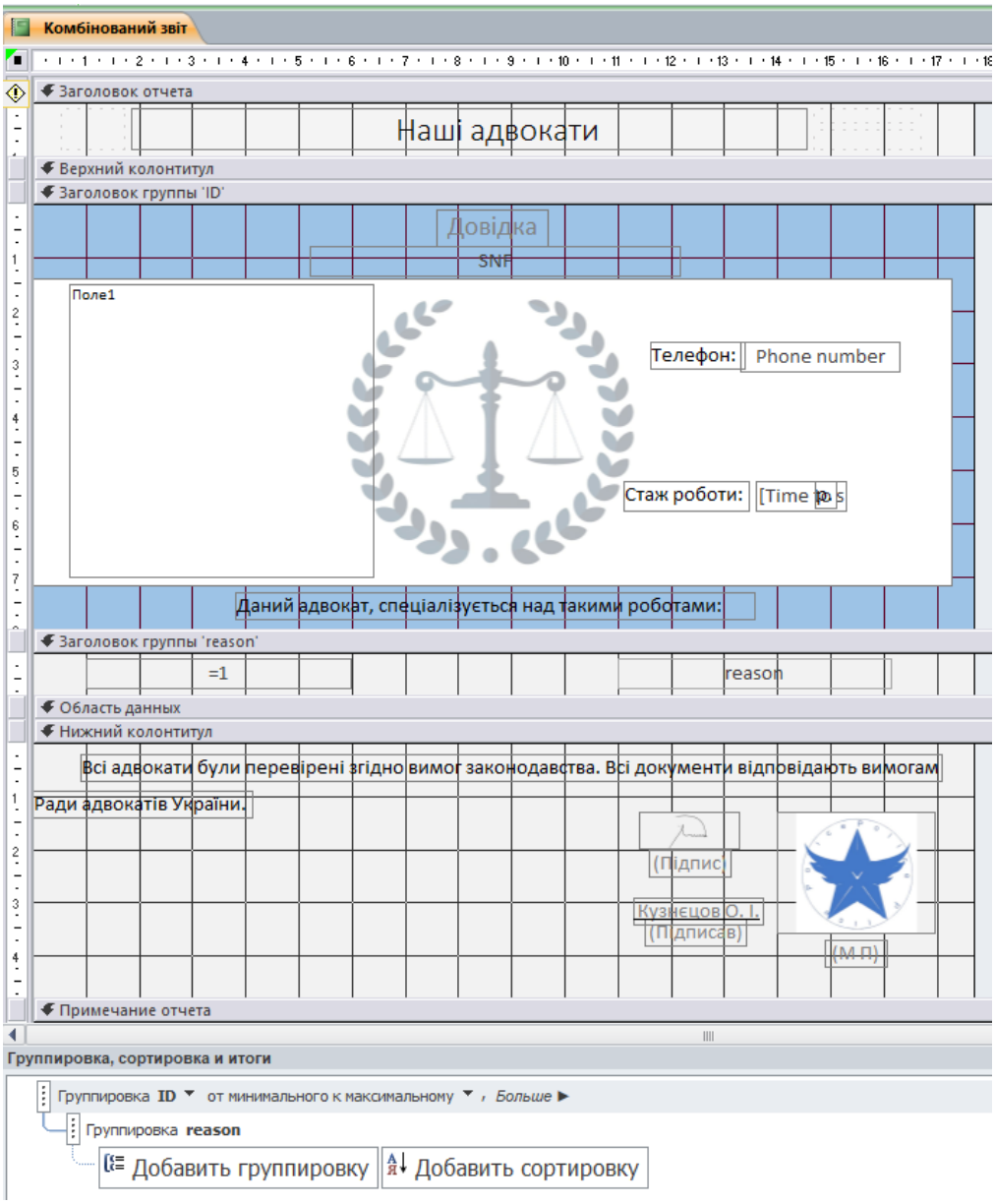


Телефон: +380502345678

Стаж роботи: 14 р.

Даний адвокат, спеціалізується над такими роботами:

1	Хуліганство
2	Шахрайство


Комбінований звіт
Заголовок отчета
Наші адвокати
Верхний колонтитул
Заголовок группы 'ID'
Довідка
SNF
Поле1
Телефон:
Phone number
Стаж роботи:
[Time
s
Даний адвокат, спеціалізується над такими роботами:
Заголовок группы 'reason'
=1
reason
Область данных
Нижний колонтитул
Всі адвокати були перевірені згідно вимог законодавства. Всі документи відповідають вимогам Ради адвокатів України.
(Підпис)
Кузнєцов О. І.
(Підписав)
(М П)
Примечание отчета
Группировка, сортировка и итоги
Группировка ID от минимального к максимальному , Больше
Группировка reason
Добавить группировку
Добавить сортировку

Лабораторна робота №5

Інтерфейс бази даних у середовищі MS Access.

ЗАВДАННЯ

1. Розробити сценарій інтерфейсу, вказавши:
   1.1.Опис набору дій, допустимих для певного користувача.
   1.2. Типи користувачів та права на виконання певних дій.
   1.3. Дії над даними у БД: добавлення записів, редагування записів, видалення записів, пошук записів.
2. Для реалізації сценарію інтерфейсу розробити необхідні форми, макроси.

МЕТОДИЧНІ ВКАЗІВКИ

1. Створити форму за допомогою автоформи.
2. Створити форму за допомогою майстра форм.
3. Створити форму вручну за допомогою конструктора форм.
4. Добавити у форму командні кнопки.
5. Переглянути форму.
6. Зберегти форму.
7. Використати форму.
8. Створити кнопочну форму.
9. Створити рядок меню, контекстне меню.
10. Створити панель інструментів.

Є два способи відображення даних із бази даних для візуального огляду:

1) у вигляді таблиці;
2) у вигляді форми.

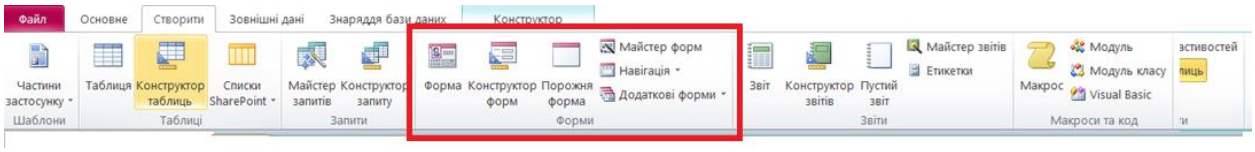
Форми дозволяють відображати дані в різноманітних форматах. Форми нагадують бланк чи картку, наприклад сторінку з паспорта, бібліографічну картку на книгу в бібліотеці. Одна форма містить дані лише з одного запису.

Використання форм для відображення даних дає додаткові можливості. Зокрема, на формі зручно розташувати:

- поля типу OLE;

- елементи керування: кнопки, перемикачі тощо;
- написи (заголовки форми/рубрик), а також розрисувати форму чи задати фоновий рисунок-заставку;
- обчислювальні поля для відображення результатів обчислень, виконаних за допомогою наявних полів;
- закладки (багатосторінкові форми, де поля групують за змістом на різних закладках).
- Є декілька способів створення форм, розглянемо деякі з них:

- атоматично, за допомогою команди **Форма**;
- за допомогою **Майстра форм**;
- вручну, за допомогою **Конструктора форм;**
- комбінованим способом.



Наприклад, виконавши команду **Форма,** отримаємо форму, в якій всі поля з таблиці будуть розташовані в стовпець і вирівняні за лівим краєм.

Користувач за допомогою конструктора може змінити розташування полів методом їх перетягування.

Використання майстра форм дає змогу швидко відібрати потрібні поля з таблиці для розміщення на формі або розташувати на одній формі поля з різних таблиць.

Для конструювання форми використовують панель елементів з кнопками (рис. 17)

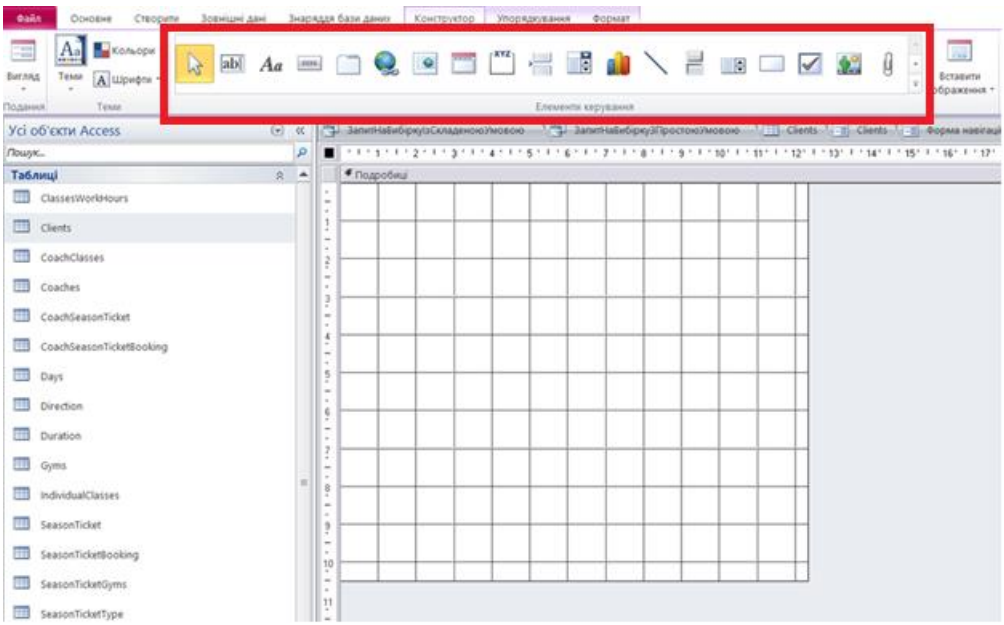


Рис. 17. Панель елементів керування конструктора форм

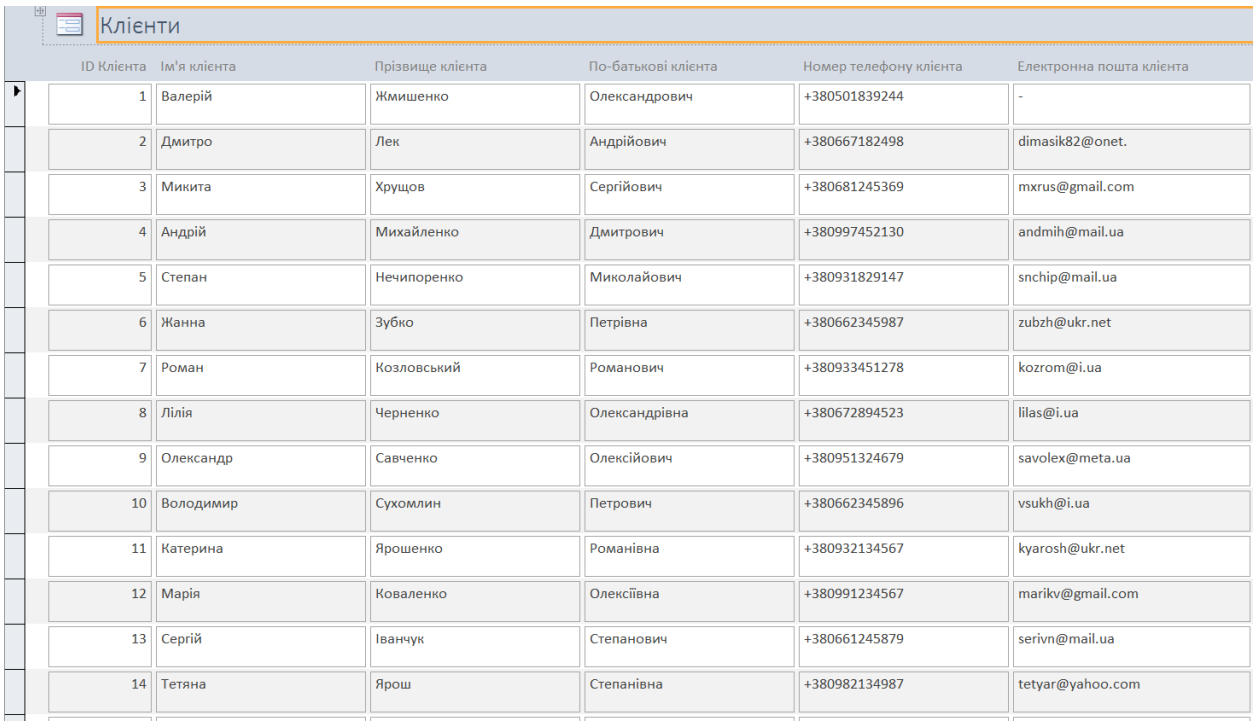
Щоб вставити елемент керування, його треба вибрати на панелі й клацнути на формі в точці вставляння.

За допомогою відповідних кнопок можна вставити у форму в діалоговому режимі інші елементи: кнопки, перемикачі тощо.

За допомогою побудованих форм можна виводити дані з таблиць або дані, відібрані згідно із запитами.

1. Створення форми автоматичним способом

При використанні команд **Форма** або **Додаткові форми** можна швидко створити форму типу в стовпець, в рядок або табличну. Вона з`являється відразу після вибору відповідної команди. Для виконання команд необхідно у вікні бланка бази даних (рис.3) вибрати вкладинку **Форми** і клікнути на кнопці **Створити**. Далі обираємо будь-який автоматичний спосіб (наприклад, **Додаткові форми**, далі вигляд форми **Кілька елементів** і одержуємо форму (рис.18).

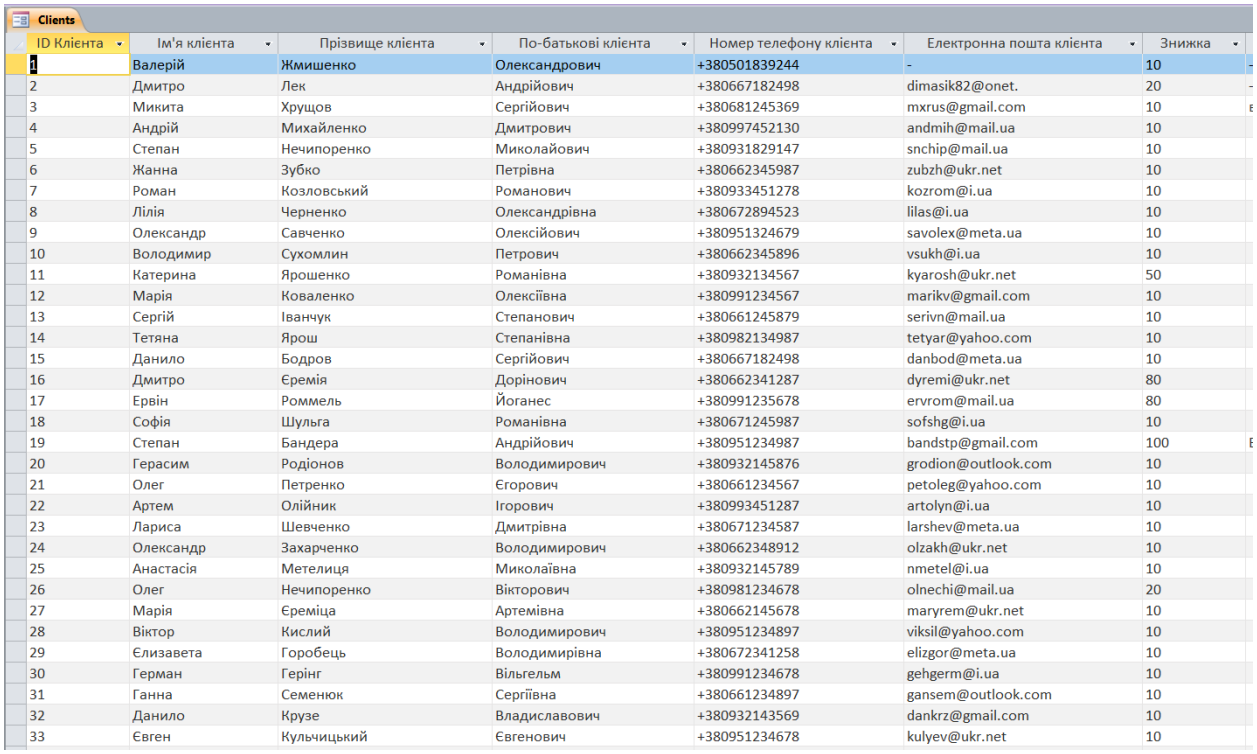


Клієнти

ID Клієнта	Ім'я клієнта	Прізвище клієнта	По-батькові клієнта	Номер телефону клієнта	Електронна пошта клієнта
1	Валерій	Жмишенко	Олександрович	+380501839244	-
2	Дмитро	Лек	Андрійович	+380667182498	dimasik82@onet.
3	Микита	Хрущов	Сергійович	+380681245369	mxrus@gmail.com
4	Андрій	Михайленко	Дмитрович	+380997452130	andmih@mail.ua
5	Степан	Нечипоренко	Миколайович	+380931829147	snchip@mail.ua
6	Жанна	Зубко	Петрівна	+380662345987	zubzh@ukr.net
7	Роман	Козловський	Романович	+380933451278	kozrom@i.ua
8	Лілія	Черненко	Олександрівна	+380672894523	lilas@i.ua
9	Олександр	Савченко	Олексійович	+380951324679	savolex@meta.ua
10	Володимир	Сухомлин	Петрович	+380662345896	vsukh@i.ua
11	Катерина	Ярошенко	Романівна	+380932134567	kyarosh@ukr.net
12	Марія	Коваленко	Олексіївна	+380991234567	marikv@gmail.com
13	Сергій	Іванчук	Степанович	+380661245879	serivn@mail.ua
14	Тетяна	Ярош	Степанівна	+380982134987	tetyar@yahoo.com

Рис.18 Вікно форми Клієнти вигляду **Кілька елементів**

Якщо обираємо автоматичний спосіб (наприклад, **Додаткові форми**, далі вигляд форми **Таблиця** і одержуємо наступну форму



Clients

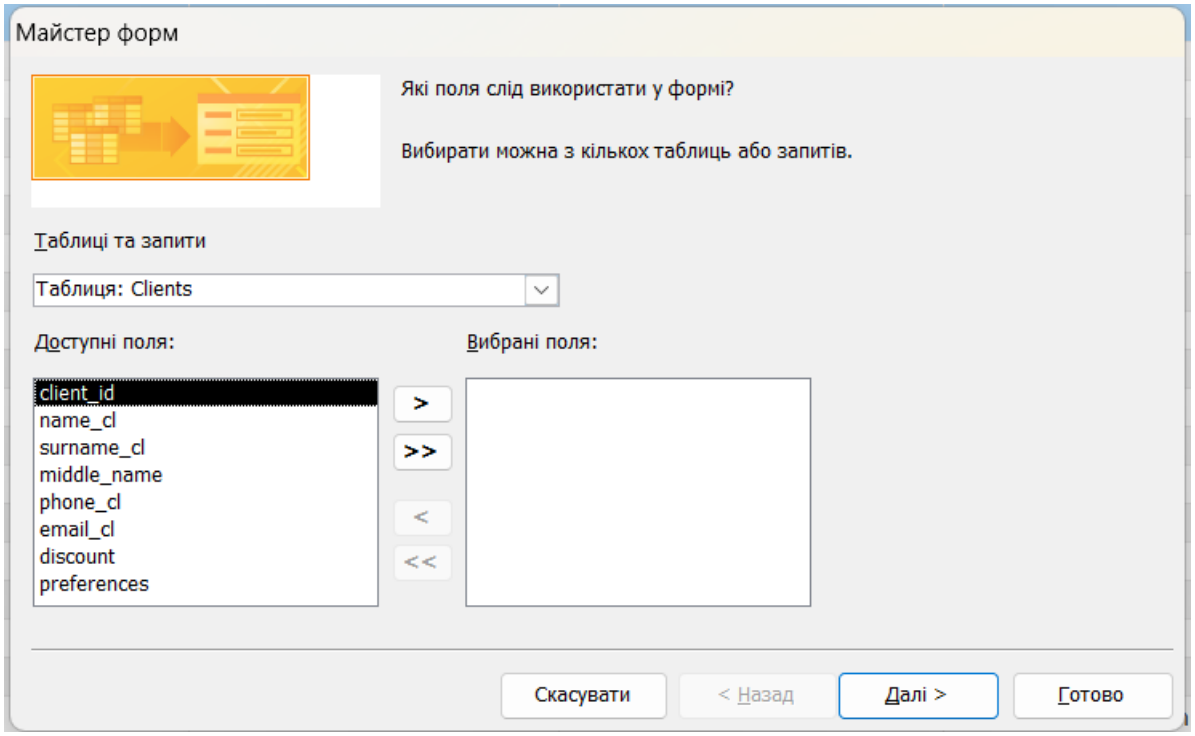
ID Клієнта	Ім'я клієнта	Прізвище клієнта	По-батькові клієнта	Номер телефону клієнта	Електронна пошта клієнта	Знижка
1	Валерій	Жмишенко	Олександрович	+380501839244	-	10
2	Дмитро	Лек	Андрійович	+380667182498	dimasik82@onet.	20
3	Микита	Хрущов	Сергійович	+380681245369	mxrus@gmail.com	10
4	Андрій	Михайленко	Дмитрович	+380997452130	andmih@mail.ua	10
5	Степан	Нечипоренко	Миколайович	+380931829147	snchip@mail.ua	10
6	Жанна	Зубко	Петрівна	+380662345987	zubzh@ukr.net	10
7	Роман	Козловський	Романович	+380933451278	kozrom@i.ua	10
8	Лілія	Черненко	Олександрівна	+380672894523	lilas@i.ua	10
9	Олександр	Савченко	Олексійович	+380951324679	savolex@meta.ua	10
10	Володимир	Сухомлин	Петрович	+380662345896	vsukh@i.ua	10
11	Катерина	Ярошенко	Романівна	+380932134567	kyarosh@ukr.net	50
12	Марія	Коваленко	Олексіївна	+380991234567	marikv@gmail.com	10
13	Сергій	Іванчук	Степанович	+380661245879	serivn@mail.ua	10
14	Тетяна	Ярош	Степанівна	+380982134987	tetyar@yahoo.com	10
15	Данило	Бодров	Сергійович	+380667182498	danbod@meta.ua	10
16	Дмитро	Єремія	Дорінович	+380662341287	dyremi@ukr.net	80
17	Ервін	Роммель	Йоганес	+380991235678	ervrom@mail.ua	80
18	Софія	Шульга	Романівна	+380671245987	sofshg@i.ua	10
19	Степан	Бандера	Андрійович	+380951234987	bandstp@gmail.com	100
20	Герасим	Родіонов	Володимирович	+380932145876	grodion@outlook.com	10
21	Олег	Петренко	Єгорович	+380661234567	petoleg@yahoo.com	10
22	Артем	Олійник	Ігорович	+380993451287	artolyn@i.ua	10
23	Лариса	Шевченко	Дмитрівна	+380671234587	larshev@meta.ua	10
24	Олександр	Захарченко	Володимирович	+380662348912	olzakh@ukr.net	10
25	Анастасія	Метелиця	Миколаївна	+380932145789	nmetel@i.ua	10
26	Олег	Нечипоренко	Вікторович	+380981234678	olnechi@mail.ua	20
27	Марія	Єреміца	Артемівна	+380662145678	maryrem@ukr.net	10
28	Віктор	Кислий	Володимирович	+380951234897	viksil@yahoo.com	10
29	Єлизавета	Горобець	Володимирівна	+380672341258	elizgor@meta.ua	10
30	Герман	Герінг	Вільгельм	+380991234678	gehgerm@i.ua	10
31	Ганна	Семенюк	Сергіївна	+380661234897	gansem@outlook.com	10
32	Данило	Крузе	Владиславович	+380932143569	dankrz@gmail.com	10
33	Євген	Кульчицький	Євгенович	+380951234678	kulyev@ukr.net	10

Спочатку активуємо назву таблиці, а потім вибираємо відповідну команду типу форми. Для перегляду всіх записів згідно із даною формою використовують кнопки прокрутки, розміщені внизу екрана виводу форми.

2.Створення форми за допомогою майстра форм

Розглянемо алгоритм створення форми в Access за допомогою **Майстра форм.**

Запускаємо **Майстра форм** на вкладці **Створення** групи **Форми.**



У вікні Майстра обираємо таблицю або запит, на основі якого створюємо форму (наприклад, Клієнти).

У списку Доступні поля вибираємо потрібні поля — натискаємо кнопку >, щоб перенести їх у список Вибрані поля.

В області відбору полів є два списки: **Доступні поля**, **Вибрані поля** і чотири кнопки:

> – добавити вибране поле;

>> – добавити всі поля;

< – вилучити вибране поле;

<< – вилучити всі поля.

У нижній частині вікна майстра форм (рис. 19) розміщені типові кнопки діалогу з майстром:

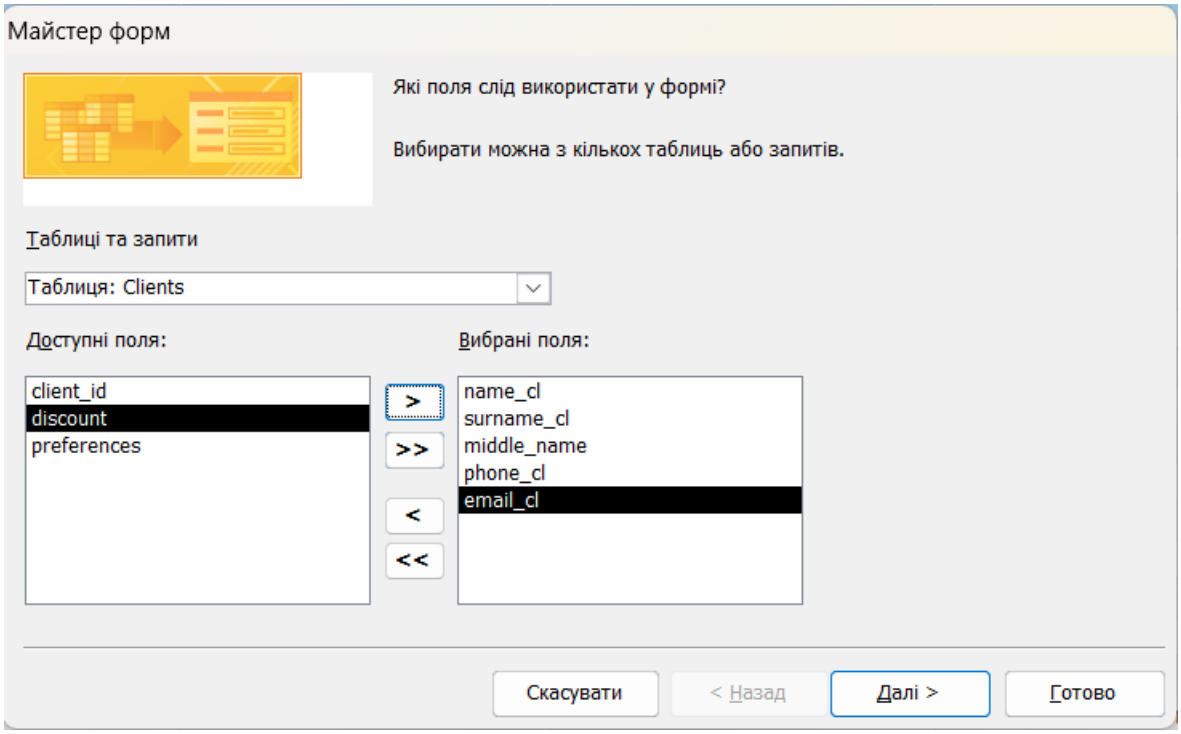


Рис. 19. Вікно відбору полів для форми

Скасувати – відміняє створення форми й дозволяє перейти в на початок;

Назад – повертає в попереднє діалогове вікно;

Далі – переходить до наступного діалогового вікна;

Готово – переходить до останнього діалогового вінка.

Після відбору полів натискуємо кнопку **Далі** і переходимо в діалог вибору зовнішнього вигляду форми. Здійснюємо вибір макета форми

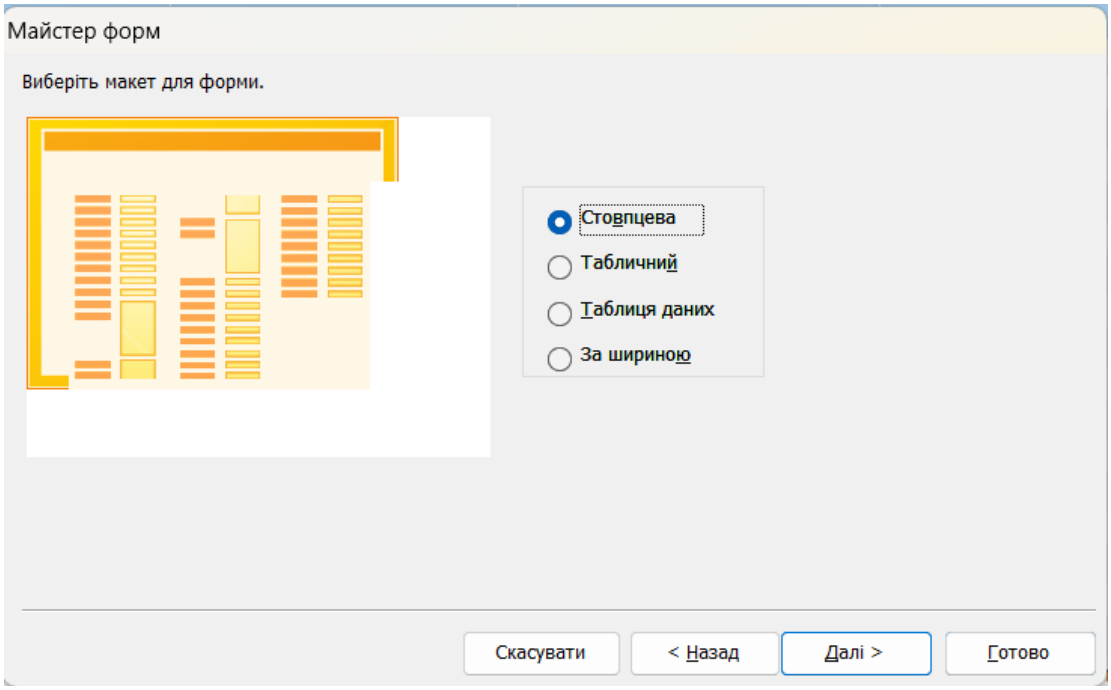


Рис. 20. Вікно вибору зовнішнього вигляду форми

Стовпцева (Columnar) — відображає один запис на екрані (зручно для введення даних).

Табличний (Tabular) — кілька записів у вигляді таблиці.

Таблиця даних (Datasheet) — схоже на таблицю Excel.

За шириною (Justified) — поля розташовані рівномірно.

Натискаємо **Далі** і обираємо стиль оформлення та завершуємо створення вказуючи назву форми (наприклад, Форма_Клієнти).

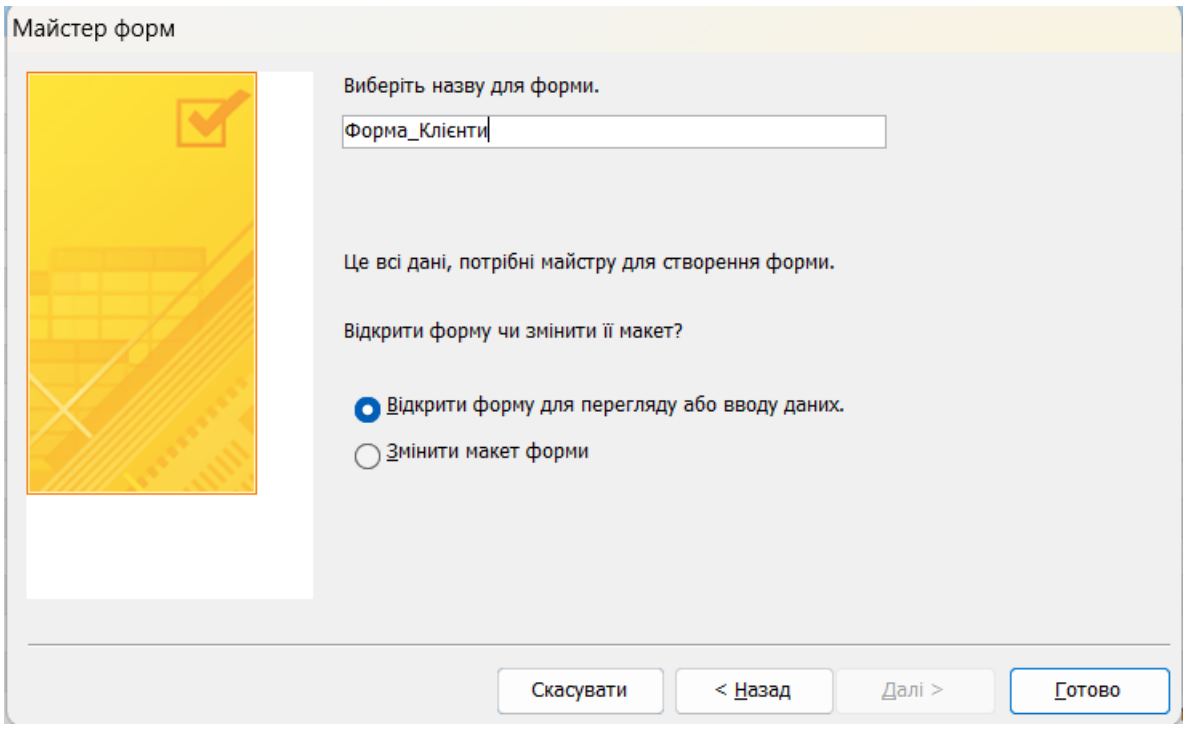
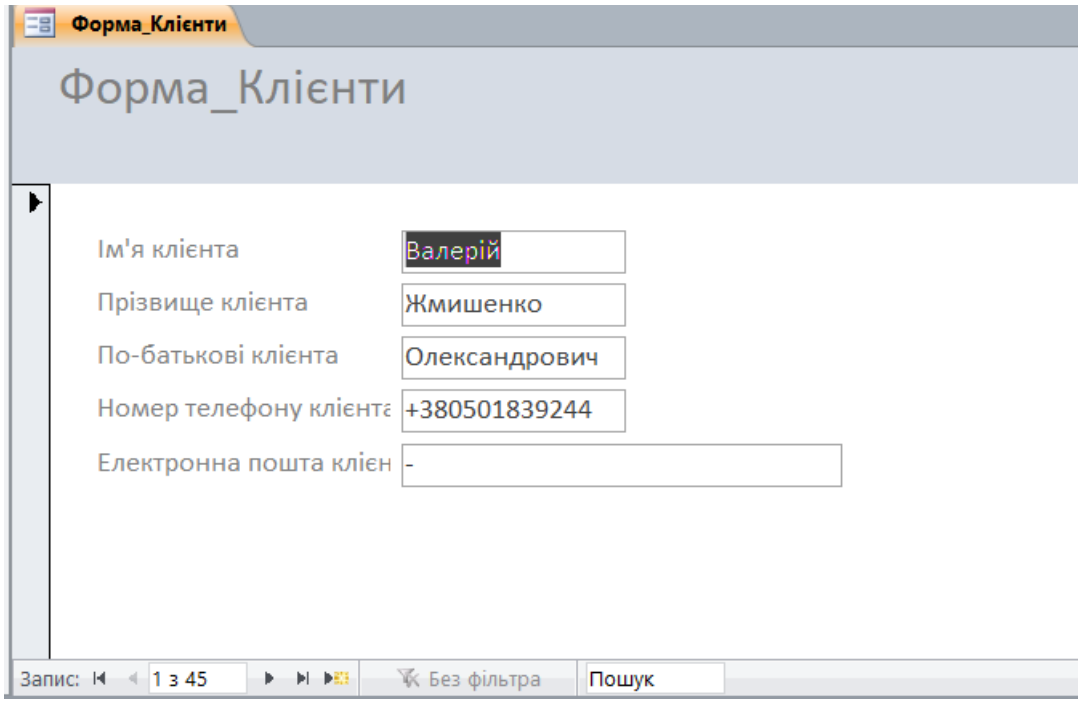


Рис. 21. Вікно введення назви та збереження форми

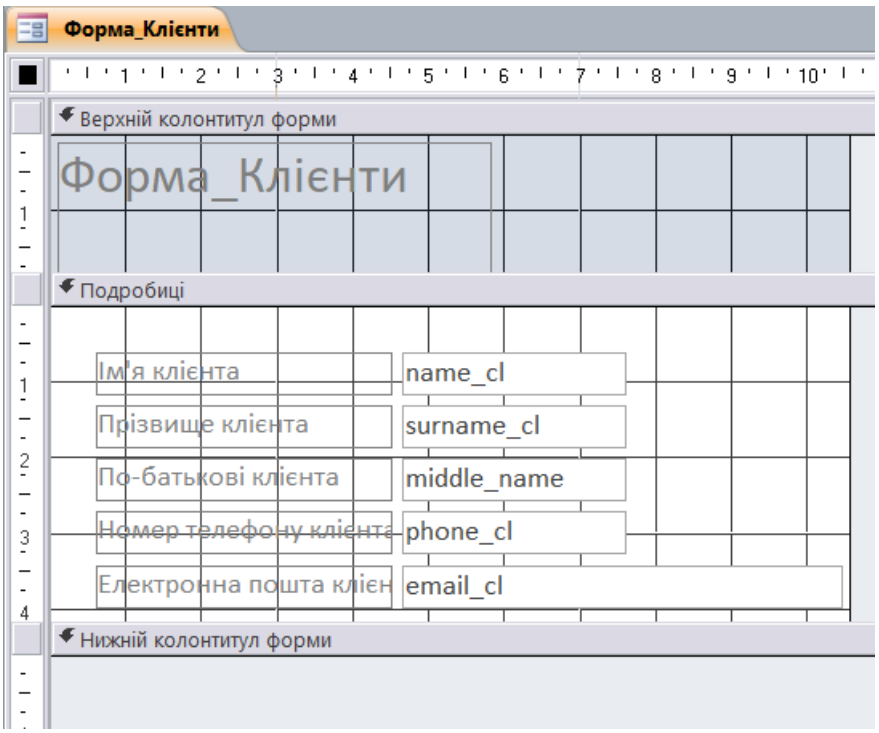
Вибираємо одну з опцій:

- Відкрити форму для перегляду або введення даних.
- Змінити макет форми (у режимі конструктора).

Натискаємо **Готово**.



Якщо потрібно змінити зовнішній вигляд, підписи або порядок полів, то переходимо у Режим **Конструктора**.



3. Створення форми вручну за допомогою конструктора форм

Форми можна розробляти вручну, відкривши порожню форму й добавивши в неї об'єкти (текст, текстові вікна, графіку, лінії, рамки і т.д.). Для створення форми вручну необхідно виконати такі дії:

1. У вікні бланка бази даних (рис. 3) вибрати вкладку **Створити**, обрати групу **Форми** і клікнути по кнопці **Конструктор форм.** На екрані відкриється порожня форма в режимі конструктора **Форма1**;

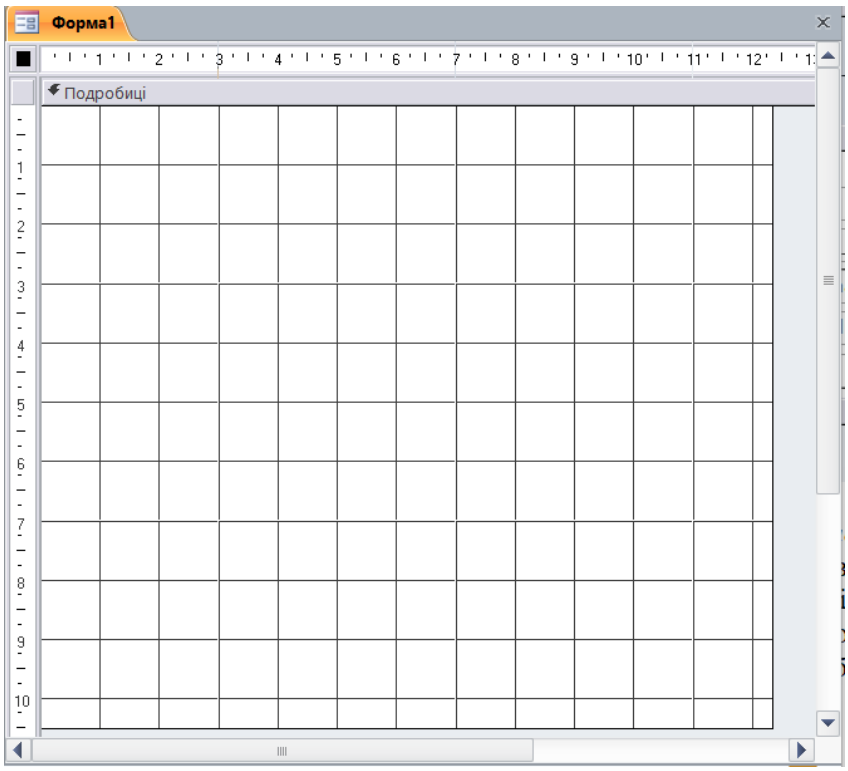
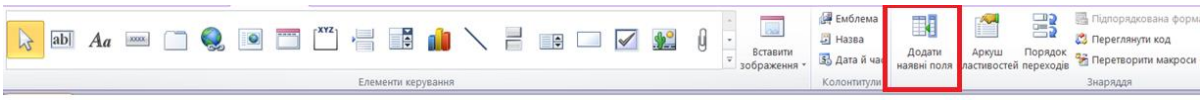
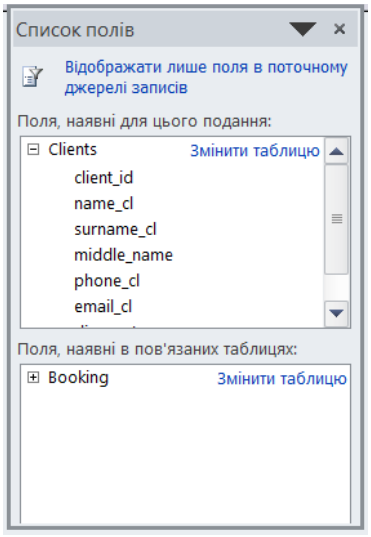


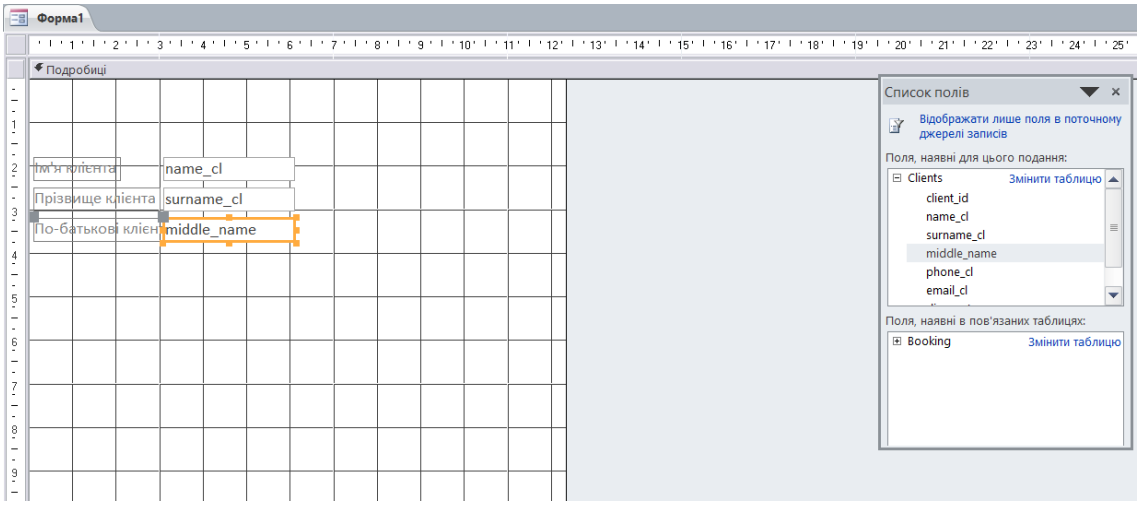
Рис. 22. Вікно порожньої форми в режимі конструктора форми

2. Вибери таблицю або запит, дані з якого використовуватимуться у формі. Для цього на панелі інструментів активуй Додати наявні поля.





3. З’являється діалогове вікно **Список полів.** У списку є усі поля обраної таблиці. Перетягни потрібні поля мишею на форму (наприклад: Прізвище, Ім’я...) і розташуй їх у зручному порядку.



Створюючи форму вручну, ми працюємо у вікні, яке можна розглядати як область для малювання, в якій ми малюємо вміст форми. Для зручності область даних розділена сіткою.

Форма може містити заголовок і примітки. Для цього необхідно виконати команду контекстного меню **Заголовки/Примітка форми.** Наприклад, щоб ввести заголовок, потрібно натиснути кнопку **Надпис** на панелі елементів і клацнути в полі заголовка в тому місці, де повинен знаходитись лівий верхній кут текстового вікна, і перетягнути вказівник мишки туди, де повинен бути правий нижній кут. Потім ввести текст заголовка. Клацнути над текстом заголовка, щоб перетворити заголовок в об`єкт. Змінити його розміри й розташування. Змінити розмір шрифту і його стиль, кольори букв, фону, меж тощо. Для зміни властивостей створюваного об`єкта необхідно на об`єкті клікнути правою клавішею мишки, із контекстного меню вибрати **Властивості**, вкладку **Макет** і задати потрібні параметри.

У такий спосіб можна змінювати властивості відображень довільних об`єктів;

3. Помістити у форму поля, текст або графіку. Для цього необхідно використати панель елементів.

Опишемо детальніше роботу з *елементами керування* – об`єктами, які поміщаються у форму (поля, текст, графіка). Елементи керування поділяються на *зв`язані, незв`язані* та *обчислювальні*.

Зв'язані елементи керування – це елементи, які прив`язуються до конкретного поля таблиці або запиту.

Незв'язані елементи керування – це елементи, які не прив`язані ні до таблиць, ні до запиту.

Обчислювальні елементи керування – це елементи, які відображають результати розрахунків на основі даних вихідної таблиці або запиту.

Найпоширенішим елементом керуванням, що поміщається у форму, є текстове вікно, яке *зв`язане* з даними поля таблиці /запиту. При переході від одного запису до іншого вміст поля з`являється всередині текстового поля. Найпростіший спосіб добавити тестове поле це перетягнути необхідне поле у визначене місце форми. На рис. 23 зображено результат таких дій.

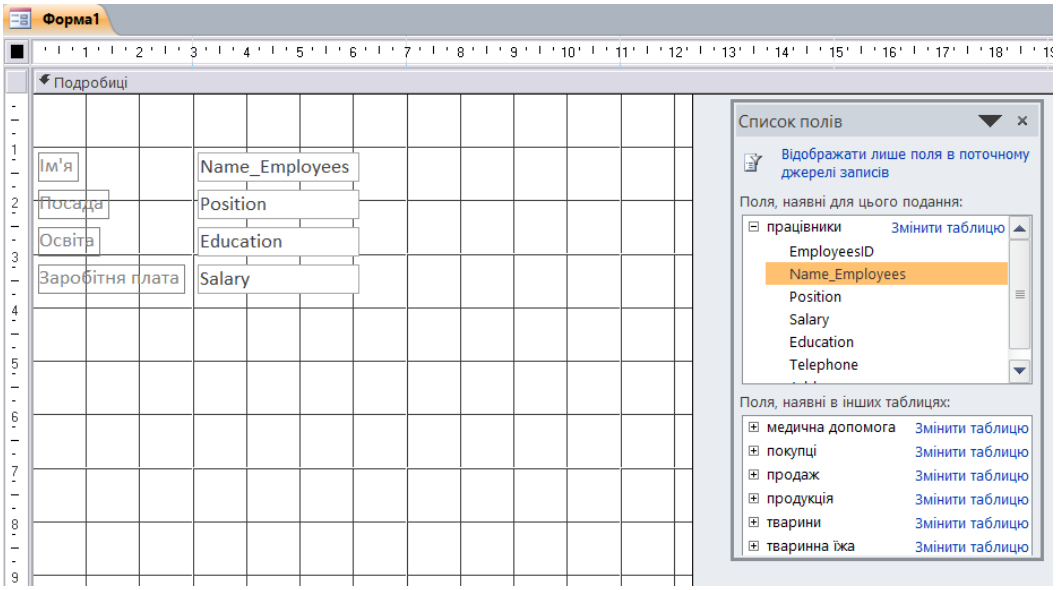


Рис. 23. Форма після перетягування в неї *зв`язаних* елементів керування

Для того, щоб помістити у форму *незв'язаний* елемент керування, необхідно вибрати на панелі елементів об'єкт **Надпис**, клікнути в тому місці форми, де повинен знаходитись лівий верхній кут текстового вікна, і перетягнути вказівник мишки туди, де повинен бути правий нижній кут. Курсор фіксується у створеному елементі, і з місця його розташування можна вводити потрібний напис.

При добавленні у форму *обчислювальних* елементів керування використовуються вирази. Наприклад, нехай необхідно підрахувати податок,

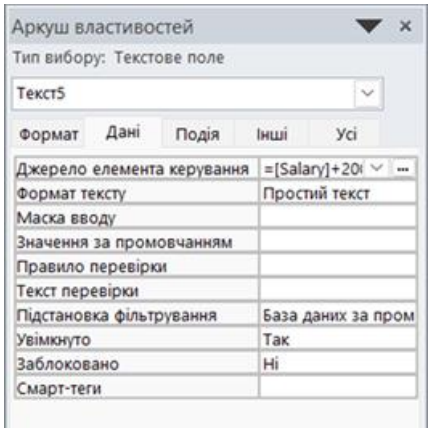
який складає 10% від величини в полі **Ціна**. Для цього потрібно виконати такі дії:

а) клацнути по кнопці **Поле** панелі елементів;

б) клацнути в тому місці форми, де повинен знаходитись лівий верхній кут текстового вікна, і перетягнути вказівник мишки туди, де повинен бути правий нижній кут;

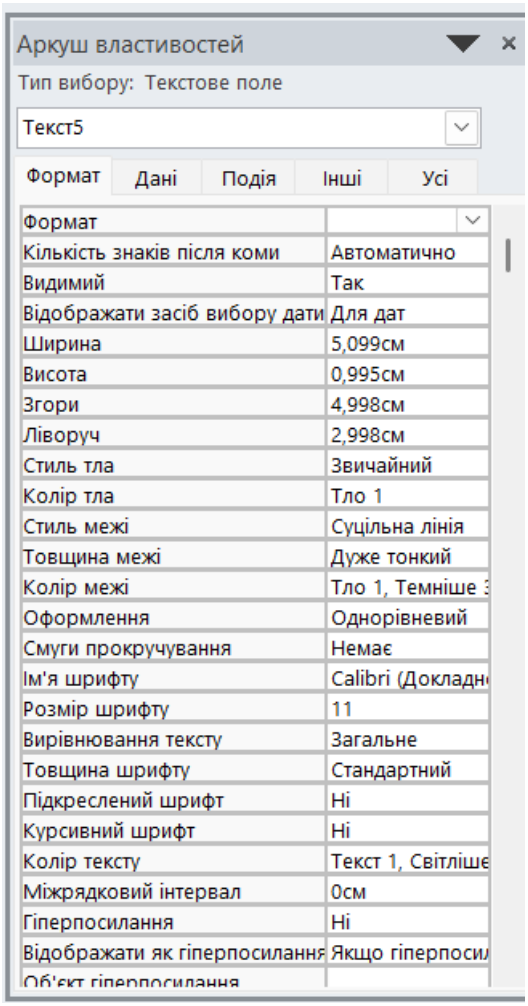
в) клацнути в текстовому вікні і ввести потрібний вираз, починаючи із знака рівності(=0.1***Ціна**).

Існує інший спосіб: клацнути правою клавішею мишки на елементі керування і вибрати, команду **Властивості** із контекстного меню, відкриється діалогове вікно **Аркуш властивостей**



далі вибираємо вкладинку **Дані** і за допомогою кнопки ... викликаємо діалог побудови виразу (рис.13).

У вікні **Властивості** можна змінити різні налаштування у вкладниці **Формат** (назву поля, шрифт, колір, вирівнювання, формат введення або підпис та ін..)



Можна також налаштувати загальні властивості форми, зокрема змінити її заголовок, фон, розміри вікна та інші параметри.

4. Добавляння командних кнопок

Кнопки – це найпростіші елементи керування форми, використовуються для запуску макросу. За допомогою кнопки, за якою закріпляються макроси, можна виконувати наступні дії:

- відкриття та відображення інших форм;
- відкриття та друк звіту;
- пошук записів та фільтрація;
- вихід з бази даних.

Макроси можна вказувати для різних властивостей кнопки: для властивості: **Натиснення кнопки** та для властивості **Подвійне натиснення кнопки**. Якщо потрібно встановити відміну реакції кнопки, то використовують макрокоманду **Відмінити Дію**.

Командні кнопки використовуються для виконання звичайних операцій керування даними, наприклад для переміщення по записах таблиці, добавлення записів, друк поточного запису тощо. Командні кнопки можна створювати за допомогою майстра командних кнопок, виконавши такі дії:

1. Відкрити форму в режимі конструктора і відобразити панель інструментів, якщо її немає на екрані;
2. На панелі інструментів клацнути по кнопці **Командна кнопка;**
3. Клацнути в тому місці форми, де необхідно помістити верхній лівий кут командної кнопки. Відкриється діалогове вікно майстра створення кнопок (рис.24);

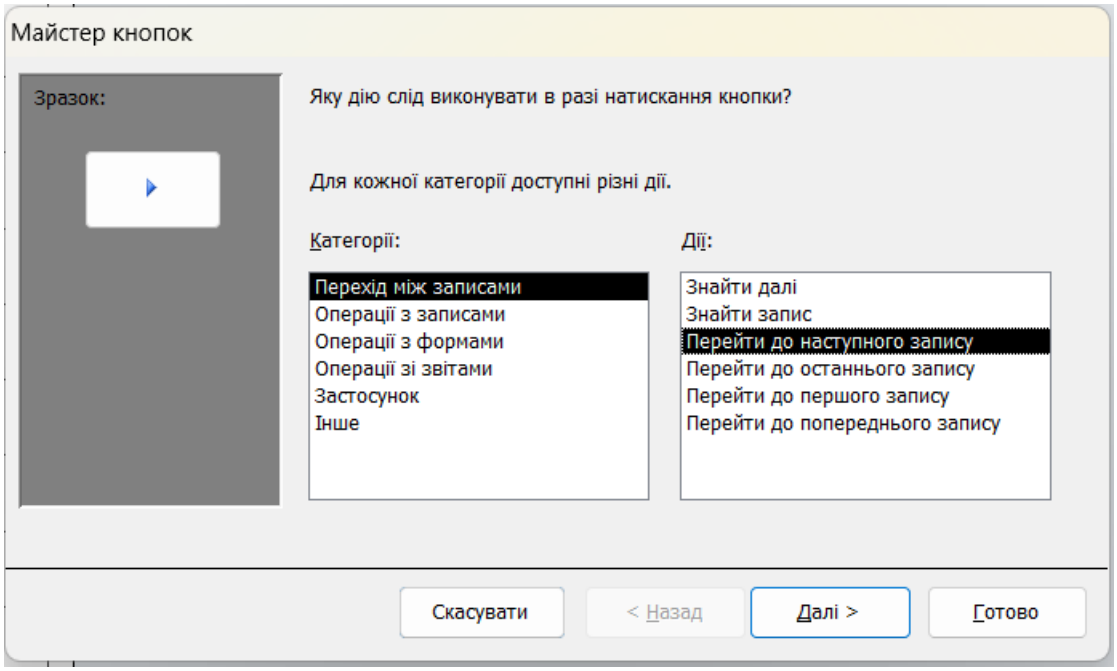


Рис. 24. Вікно **Майстер кнопок**

4. У вікні **Майстер кнопок** потрібно вибрати одну із категорій кнопки. В кожній категорії є декілька дій. Список дій змінюються згідно із обраною категорією. Якщо для кнопки вибрана категорія роботи із формами, звітами, запитами чи іншими елементами бази даних, то майстер відкриє вікно вибору імені потрібного об’єкту. Доступними будуть всі елементи відповідного типу. Наприклад, **Перехід між записами** і дію **Перейти до наступного запису**, перейти в наступне діалогове вікно через кнопку **Далі** (рис. 25);

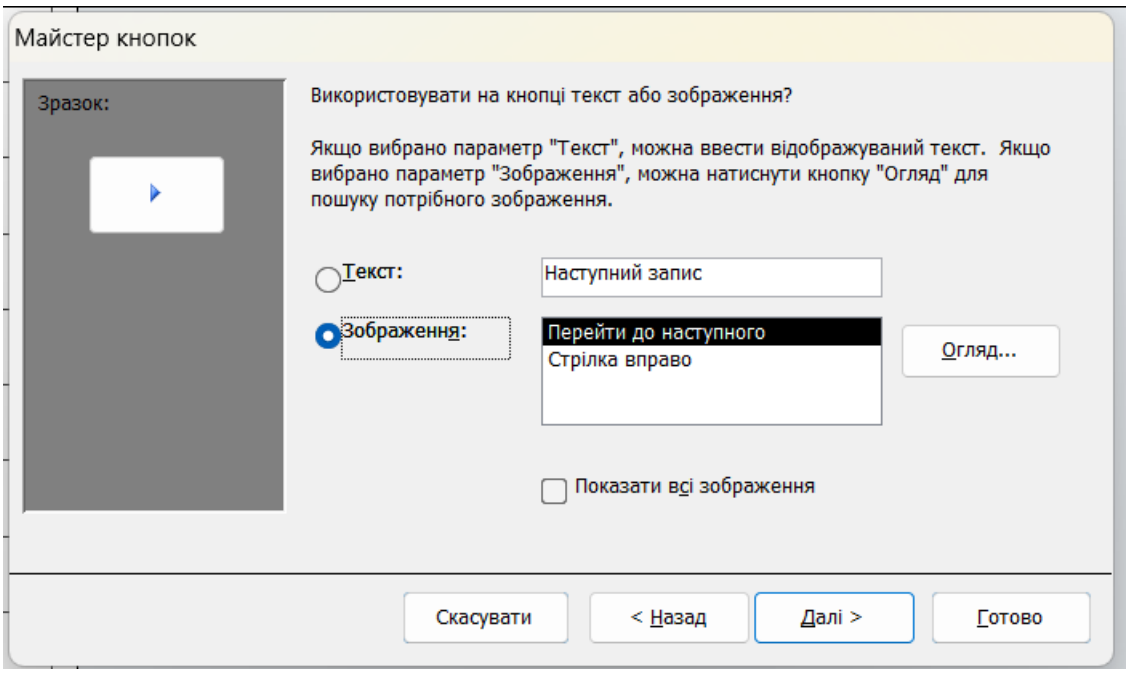
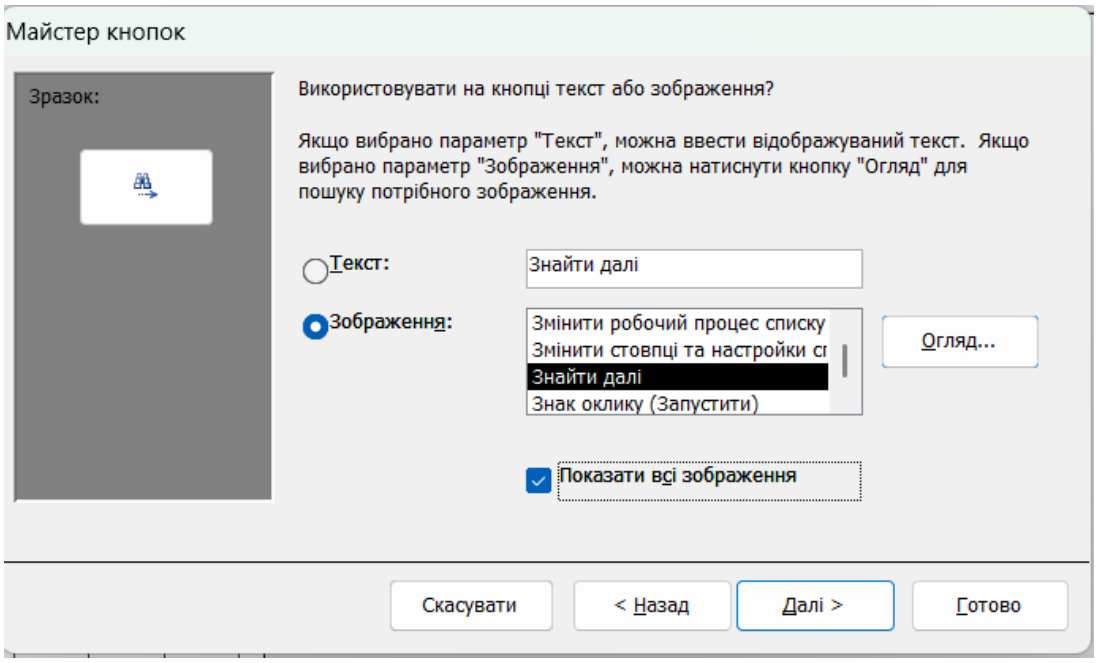
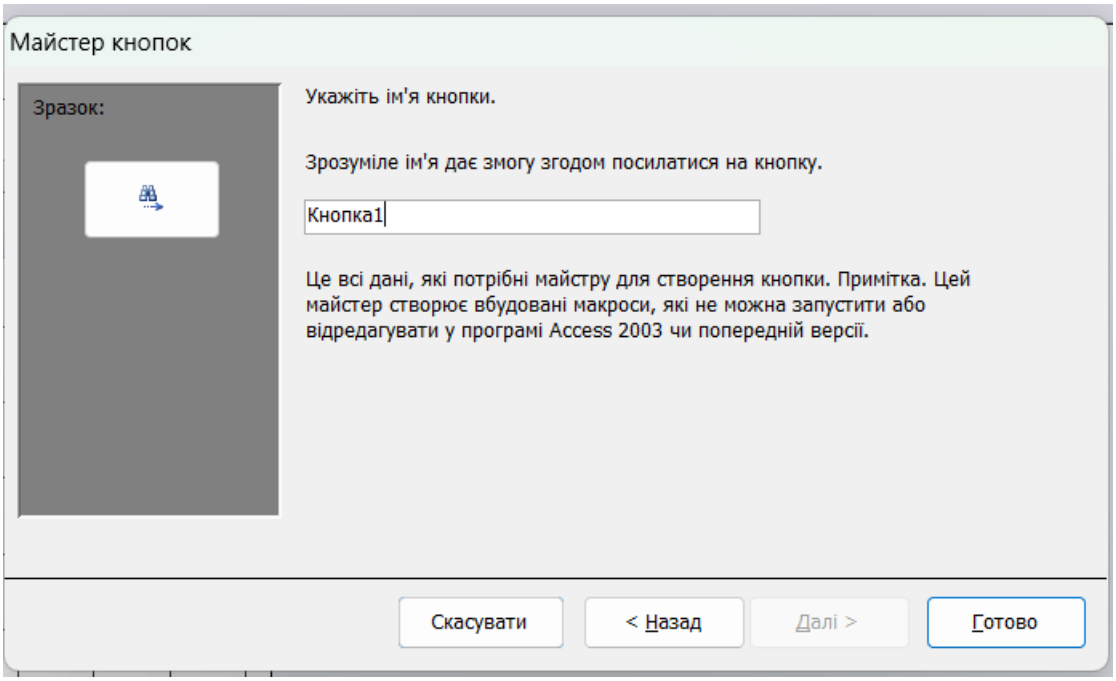


Рис.25. Вікно задання тексту або малюнка кнопки

5. У другому вікні майстра елементів керування потрібно визначити спосіб оформлення кнопки: відображати на кнопці напис чи малюнок. Якщо обрано виведення тексту на кнопці, то вводять потрібне із клавіатури. При виборі малюнка, у вікні відображаються лише тематичні малюнки. Можна переглянути всі малюнки, які має майстер, встановивши опцію **Показати всі зображення**.



6. Третє вікно роботи майстра потребує введення імені кнопки. Це внутрішнє ім’я кнопки, яке використовується конструктором.



7. Робота майстра завершується після натиснення кнопки **Готово**.

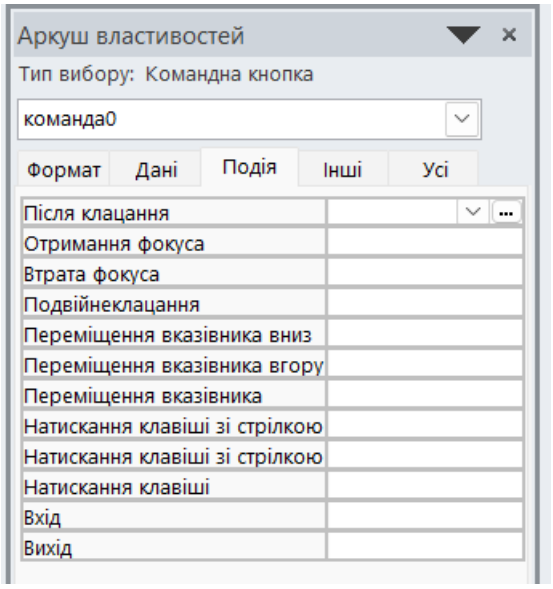
Зауваження. У властивості кнопки **Натискання кнопки**, створеної за допомогою майстра, буде зазначено **Процедура обробка події**. Це означає, що даній кнопці відповідає модуль програми поза формою, у коді VBA. Щоб подивитися його, натискають кнопку ....

Для створення кнопки за допомогою *перетягування* виконують такі дії:

1. Відкривають форму в режимі конструктора і розташовують вікно конструктора так, щоб було видно вікно бланка бази даних.
2. У вікні бланка бази даних переходять на вкладинку **Макроси** і виділяють ім’я потрібного макроса.
3. За допомогою натисненої лівої клавіші миші перетягують виділений макрос у форму. У вікні форми появиться нова кнопка.
4. При відкритті властивостей кнопки (наприклад, через контекстне меню, командою **Властивості**), потрібно перевірити чи вказано ім’я раніше вибраного макроса у властивості **Натискання клавіші**.

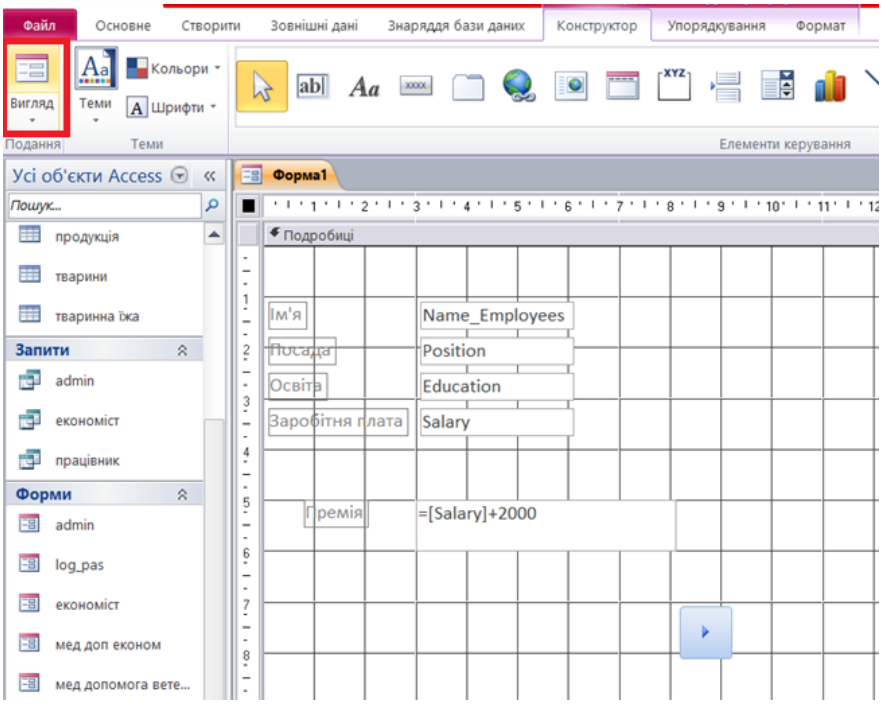
Зауваження. Кнопки створювати можна, використовуючи лише елемент керування **Кнопка** панелі інструментів вікна конструктора форм. В цьому

випадку налаштовують властивості кнопки **Натискання клавіші** і, якщо потрібно **Подвійнеклацання** кнопки.

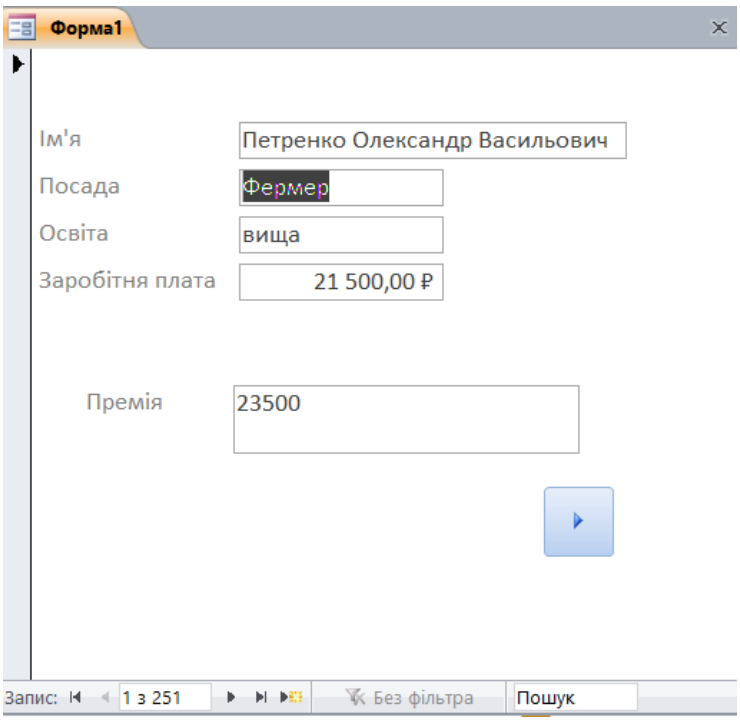


5. Попередній перегляд форми

На будь-якій стадії розробки форми її можна переглянути.

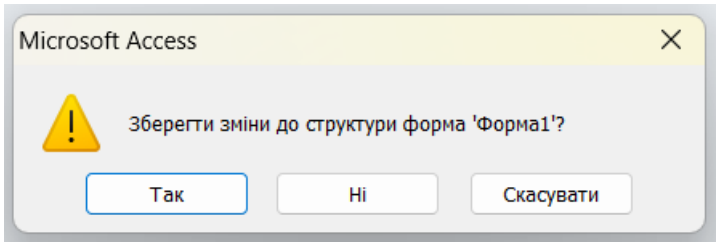


Для цього необхідно обрати режим подання **Вигляд** на панелі інструментів і одержуємо наступне подання форми.

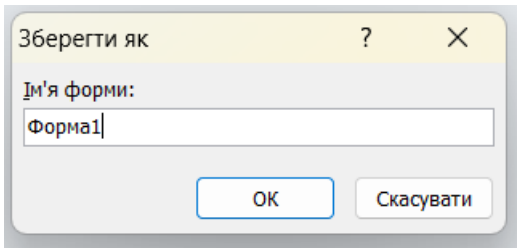


6. Збереження форми

Для збереження форми необхідно виконати команду **Зберегти** із меню **Файл** або закрити вікно форми і в діалоговому вікні, що з`являється, клацнути на кнопці **ОК.**



Якщо форма зберігається вперше, то відобразиться діалогове вікно, в якому потрібно задати ім’я форми.



7.Використання форми

Щоб відкрити збережену форму та перейти до роботи з даними, потрібно двічі клацнути лівою кнопкою миші на її піктограмі у вікні бланка бази даних. Після цього форма автоматично завантажиться та відобразиться на екрані.

Альтернативним способом є попередній вибір потрібної форми зі списку доступних об’єктів, після чого слід натиснути кнопку **Відкрити** на панелі інструментів. Обидва методи дозволяють швидко отримати доступ до форми та розпочати її використання відповідно до поставлених завдань.

8. Створення кнопочну форму

Кнопочна форма – це створена користувачем форма з кнопками. За допомогою цих кнопок можна, наприклад, запускати макроси, відкриття форми, можна об’єднувати створені раніше об’єкти бази даних. Кнопочна форма використовується в основному в якості головного меню бази даних.

Для створення кнопочної форми добавляють кнопки в довільну форму. Оскільки кнопочна форма виконує тільки роль меню бази даних, то в ній слід використовувати найменше елементи керування. Тому кнопочна форма, як правило, містить тільки кнопки, написи, OLE-об’єкти, рисунки, лінії, прямокутники.

9. Створення макросів

Макрос – це такий же об`єкт ACCESS, як і таблиця, запит чи форма. Він використовується для автоматизації частовживаних дій у базі даних. Макрос може виконувати одну дію або декілька. Довільна команда макроса називається макрокомандою. Їх більше 50. Макроси створюються у вікні макросів. Макроси можна використовувати для виконання наступних дій:

- зв’язування або сумісний запуск запитів і звітів;
- одночасне відкриття декількох форм і/ чи звітів;
- контроль введення даних при заповненні форми;
- переміщення даних між таблицями;
- виконання дій при натисненні кнопки макрокоманди.

Як і при роботі з іншими об`єктами Access, для створення макросів використовується спеціальне вікно конструктора. Щоб його відкрити потрібно виконати наступні дії.

У вікні бланка бази даних перейти на вкладку **Макроси та код** натиснути кнопку **Макрос**. Відкриється вікно конструктора макросів, у верхній частині якого послідовно вказуються макрокоманди, а в нижній частині – аргументи макрокоманди (рис.36).

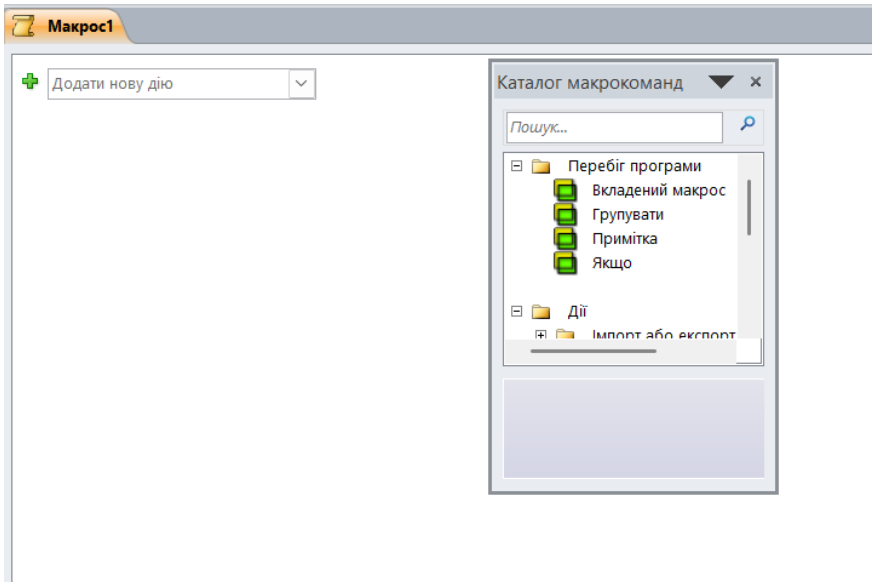
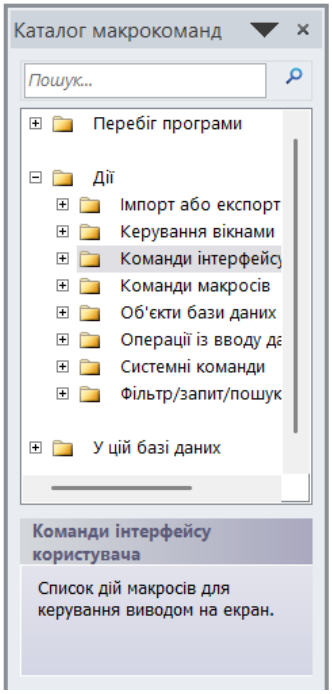


Рис 36. Вікно конструктора макросів

Послідовно вводять макрокоманди. Причому введення можливе із клавіатури чи вибором із списку, що розкривається у стовпчику **Макрокоманда**.



У цьому списку макрокоманди вказуються за групами. Імена макрокоманди вказуються українською мовою. Частина з них потребує вказання аргументів. Макрокоманди можна вказувати, перетягуючи потрібні об’єкти із вікна

бланка бази даних у відповідні комірки макрокоманд. Цей метод зручний, якщо потрібно вказати макрокоманду відкриття конкретної форми, запиту чи звіту.

В одному макросі можна вказати довільну кількість макрокоманд. Можна також створювати групу макросів, відділяючи у вікні конструктора макросів один від одного лише вказанням імені (рис.37).

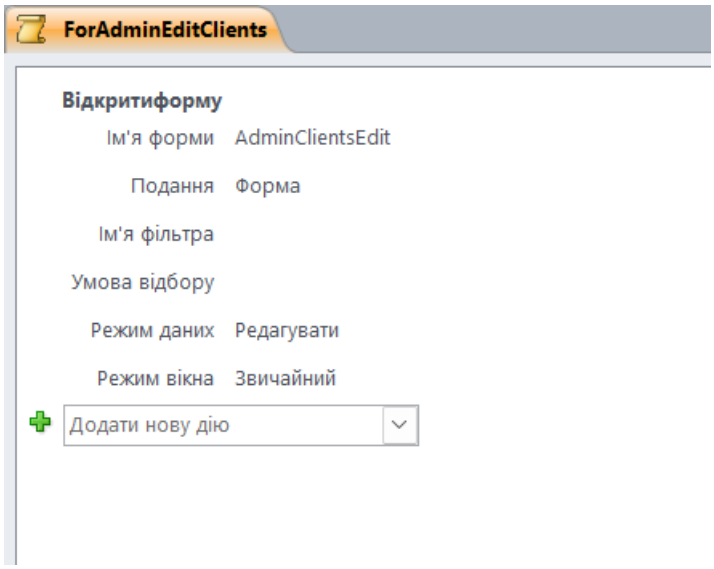


Рис. 37. Вікно конструктора макросів із групою макросів.

Для збереження макросів закривають вікно конструктора і вказують ім’я макросу. Якщо створювали групу макросів, то вказане ім’я буде іменем групи макросів і звернення до макросу групи відбуватиметься так: *Ім’я групи макросу.ім’я макросу*.

Як і інші об’єкти бази даних, макроси можна редагувати, перейменовувати, копіювати, знищувати.

10. Використання макросів

Виконати макрос можна:

- із вікна макросів;
- із вікна бланка бази даних;
- із інших вікон бази даних;
- із інших макросів;
- при виникненні події у вікні (натиснення кнопки у формі, подвійне натиснення кнопки у формі, відкриття форми, звіту, запита).

Для виконання макросу із вікна макросів достатньо натиснути кнопку **Запуск** .

Для виконання макросу у іншому макросі, вказують макрокоманду **ЗапуститиМакрос**, а в його аргументах вказують ім'я макросу.

Для активізації макросу при виникненні певної події, ім'я його вказують у відповідній властивості об'єкта, для якого можлива певна подія. Наприклад, для елемента керування **Кнопка** форми можлива подія **Натиснення кнопки**. Однойменна властивість є у вікні властивостей. Якщо вказати ім'я макросу, то він буде виконуватися при натисненні кнопки. Макроси можна також виконувати при виникненні події над формою чи іншим вікном бази даних.

11. Створення рядка меню, контекстного меню.

Крім кнопочної форми в Access можна створювати спеціальне меню, за допомогою якого можна розширити можливості системи, що розробляється. В це меню можна добавити команди, що часто використовуються, а також ті, які вже задані для кнопок форми. Щоб створити спеціальне меню, потрібно виконати такі дії:

1. Створити для кожного пункту меню групу макросів. Кожен макрос групи буде реалізовувати окрему команду пункту меню.

2. Створити новий макрос, який мав би макрокоманду **Добавити меню**. Потрібно вказати аргументи цієї макрокоманди, визначивши назву групу макросів. Якщо меню містить декілька пунктів, то для кожного пункту вказують макрокоманду **Добавити меню** відповідними аргументами.

3. Відкривають потрібну форму у режиму конструктора і у властивості форми **Рядок меню** вказують назву останнього макросу.

Доповнити проект можна також створивши контекстне меню. Його можна задавати для всієї форми або для кожного окремого елемента керування. Контекстне меню створюється так само як і стандартний рядок меню. Воно складається із контейнера команд на першому рівні і, власне команд меню на другому рівні. Для визначення контекстного меню форми чи довільного

елемента керування ім`я відповідного макросу вказують у властивості **Контекстного меню**. При натисненні правої кнопки миші в режимі перегляду форми, стандартне контекстне меню замінятиметься створеним.

Як створити контекстне меню в Access:

1. Відкрийте Конструктор макросів

У вікні бази даних перейдіть на вкладку **Створення**. Виберіть **Макрос**.

Відкриється вікно конструктора, де ви зможете створити нове контекстне меню.

2. Створіть групу макросів для контекстного меню

Щоб контекстне меню працювало, потрібно створити **групу макросів**, у якій кожен макрос є окремим пунктом меню.

Наприклад:

- Відкрити форму
- Оновити запис
- Видалити запис
- Показати повідомлення

У вікні конструктора:

1. У стовпці **Ім’я макросу** впишіть назви пунктів меню.
2. У стовпці **Макрокоманда** задайте дію (наприклад: *OpenForm*, *RunCommand*, *MessageBox*).

Після цього збережіть макрос, наприклад під назвою **mnuМоєМеню**.

3. Створіть макрос автоконтекстного меню

Далі потрібно створити спеціальний макрос, який міститиме лише одну команду — **КонструкторМеню**.

- Створіть новий макрос.
- У полі **Макрокоманда** виберіть **КонструкторМеню** (CreateMenu/ AddMenu).
- У властивостях вкажіть:
  - **Ім’я меню:** будь-яке, наприклад *Моє меню*
  - **Макроси групи:** ім’я групи макросів, наприклад *mnuМоєМеню*

Збережіть його як **AutoKeys** або з будь-яким іншим ім’ям, якщо запускатимете вручну.

4. Призначте меню формі або елементу керування

Щоб контекстне меню працювало:

- Відкрийте форму в режимі конструктора.
- Виберіть форму або потрібний елемент керування.
- У вікні властивостей знайдіть:
    - **Показувати контекстне меню** – встановіть *Так*
    - **Контекстне меню (Shortcut Menu Bar)** – впишіть ім’я створеного меню, наприклад:
      mnuМоєМеню

5. Збережіть та протестуйте

   Перейдіть у режим форми. Клацніть правою кнопкою — має з’явитися створене вами меню.

12. Створення навігаційної форми

Розглянемо як створити навігаційну форму в Access
1. Відкрийте вкладку **Створення.** У головному вікні Access перейдіть на вкладку **Створення**.
2. Виберіть **Навігаційна форма** у групі **Форми** натисніть кнопку **Навігація**.

Access запропонує кілька варіантів:

- Проста навігаційна форма
- Навігація з вкладками
- Горизонтальна / вертикальна навігація

Виберіть той тип, який вам підходить.

3. Додайте об’єкти до навігаційної панелі

Після створення форми:

1. У вікні **Об’єкти бази даних** знайдіть форми, звіти або запити.
2. Захопіть потрібний об’єкт мишею.
3. Перетягніть його у область навігації у новоствореній формі.

Access автоматично створить кнопку, яка відкриватиме вибраний об’єкт.

4. Налаштуйте кнопки навігації

За потреби можна:

- змінити назви кнопок;
- встановити свій текст чи піктограми;
- додати додаткові кнопки (збереження, закриття, друку тощо);
- змінити колір і стиль навігаційних елементів через вкладку **Форматування**.

5. Збережіть навігаційну форму

Натисніть **Ctrl + S**, введіть назву, наприклад: **frmНавігація**

Тепер це буде основна форма для переходу між елементами бази. Щоб навігаційна форма відкривалася автоматично:

1. Перейдіть у **Файл → Параметри**.
2. Виберіть **Поточна база даних**.
3. У полі **Відображати форму** оберіть свою навігаційну форму.

Лабораторна робота №5

Розробка бази даних у середовищі mySQL

ЗАВДАННЯ

1. Встановити СУБД MySQL, включаючи MySQL Shell, MySQL Workbench, MySQL Command Line Client.

2. Створити базу даних своєї предметної області, наповнивши її структурами таблиць, відповідно до розробленої у лабораторній роботі №1 схеми бази даних.

3. Заповнити таблиці даними, близькими до реальних. Усі завдання виконати у одному із засобів спілкування клієнта, використовуючи конструкції мови SQL.

МЕТОДИЧНІ ВКАЗІВКИ

MySQL є системою управління реляційними базами даних (СУБД). На сьогодні це одна з найпопулярніших систем управління базами даних.

Первинним розробником цієї СУБД була шведська компанія MySQL AB. У 1995 році вона випустила перший реліз MySQL. У 2008 році компанію MySQL AB було куплено компанією Sun Microsystems, а в 2010 році вже компанія Oracle поглинула Sun і тим самим набула прав на торгову марку MySQL. Тому MySQL на сьогодні розвивається під егідою Oracle.

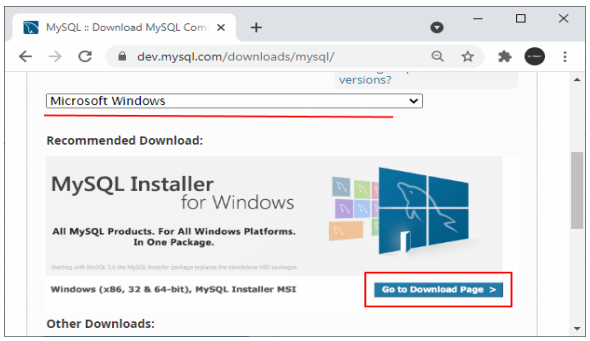
Поточною актуальною версією СУБД є версія 8.0, яка вийшла у січні 2018 року, але для якої постійно виходять підверсії На даний момент вже видані версії 8.4.7 та 9.4.0 доступні для скачування.

MySQL володіє кроссплатформенністю, існують дистрибутиви для найрізноманітніших ОС, зокрема найпопулярніших версій Linux, Windows, MacOS.

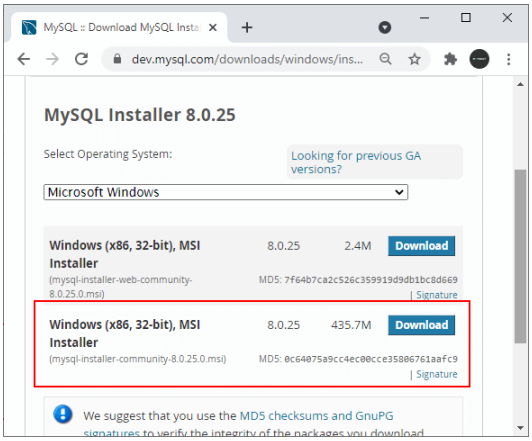
Офіційний сайт проекту: https://www.mysql.com/.

1. Встановлення MySQL

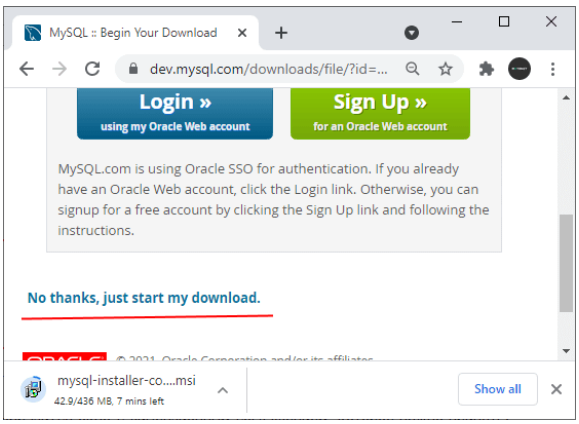
Для встановлення MySQL завантажимо дистрибутив за адресою http://dev.mysql.com/downloads/mysql/ та оберемо потрібну версію.



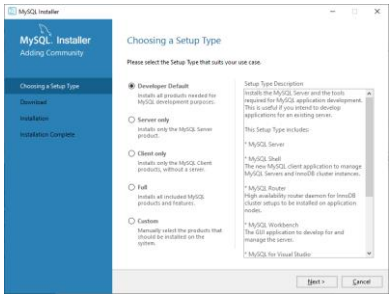
Після вибору версії натиснемо на кнопку "Go to Download Page", і нас перенаправити на сторінку завантаження дистрибутива. Тут можна обрати або онлайн-завантажувач, або повний пакет інсталятора. Можна обрати будь-який:



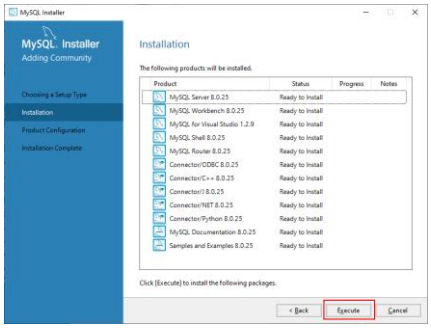
Далі може бути запропоновано залогінитися за допомогою облікового запису Oracle. Можна пропустити все це і без будь-якого логіну натиснути на посилання "No thanks, just start my download.", і почнеться завантаження:



Спочатку буде запропоновано обрати тип встановлення. Оберемо тип **Developer Default**, якого цілком вистачить для базових потреб, і натиснемо на кнопку Next:

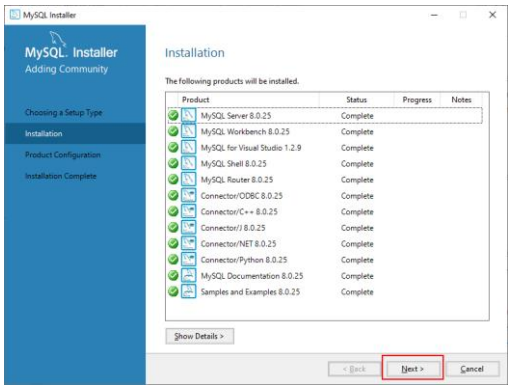


Потім на етапі встановлення інсталятор відобразить весь список компонентів, що встановлюються. У мене він виглядає так:

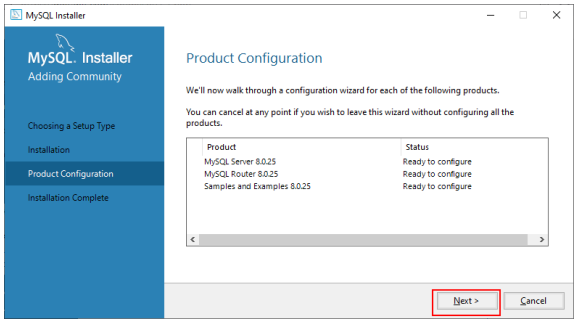


Щоб виконати встановлення всіх компонентів, натиснемо кнопку Execute.

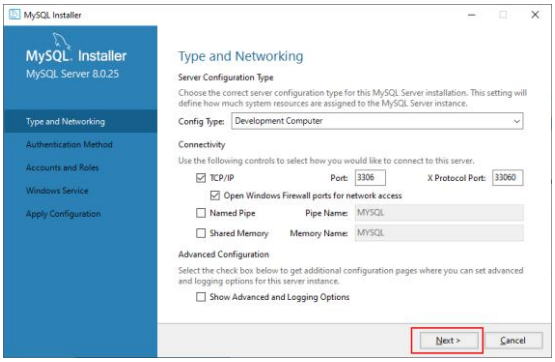
Після того, як усі компоненти будуть встановлені, натиснемо кнопку Next.



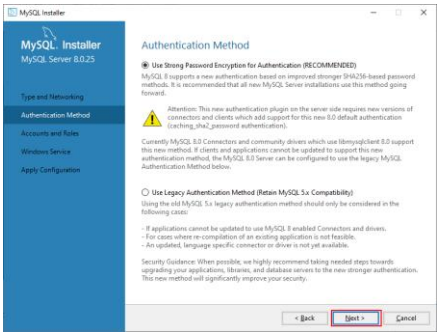
Далі відобразиться вікно з переліком продуктів, готових до конфігурації.



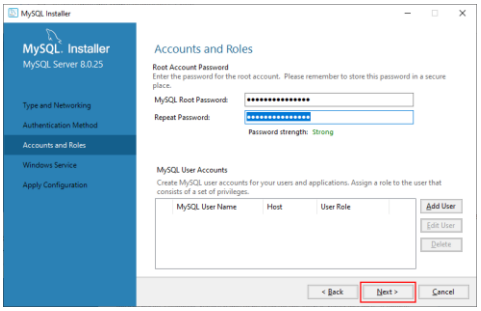
Натиснемо на кнопку Next і далі буде запропоновано встановити ряд конфігураційних налаштувань сервера MySQL. Зокрема, тут ми бачимо, що для підключення застосовуватиметься протокол TCP/IP та порт 3306. Залишимо усі ці налаштування з'єднання та порту за замовчуванням:



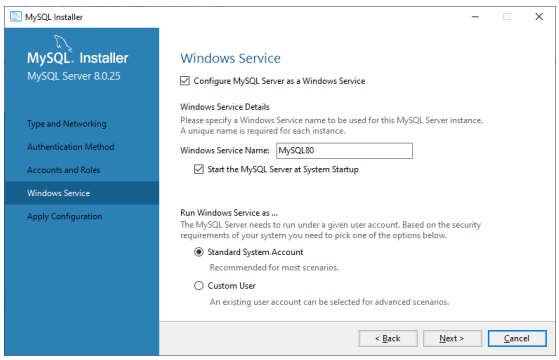
На наступному кроці буде запропоновано встановити метод аутентифікації. Залишимо налаштування за замовчуванням:



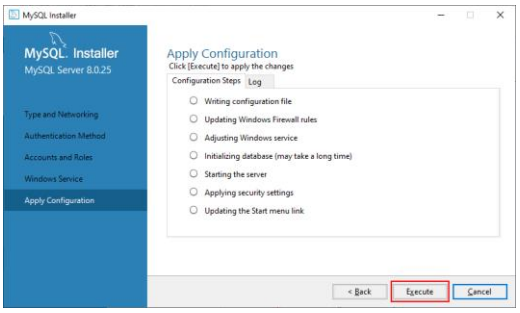
Потім у наступному вікні програми встановлення вкажемо який-небудь пароль і запам'ятаємо його, оскільки він потім знадобиться при підключенні до сервера MySQL:



Наступний набір конфігурацій, який також залишимо за замовчуванням, вказує, що сервер запускатиметься як служба Windows при запуску операційної системи:



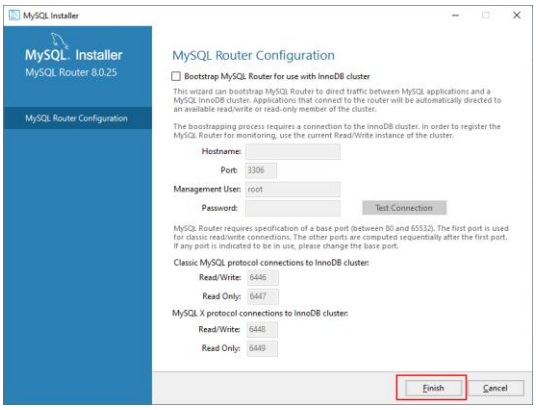
І на наступному екрані необхідно застосувати всі раніше встановлені конфігураційні налаштування, натиснувши на кнопку Execute:



Після застосування конфігураційних налаштувань сервер MySQL буде повністю встановлено і сконфігуровано, натиснемо на кнопку **Finish**.

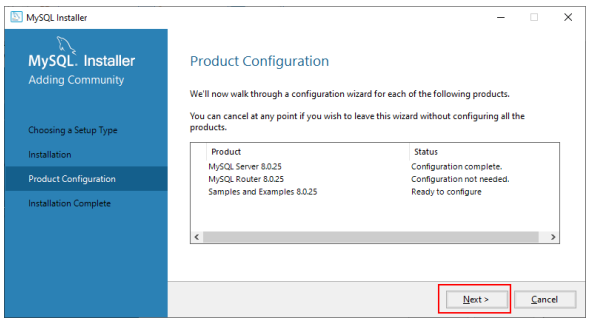
Далі знову відобразиться вікно з переліком продуктів, готових до конфігурації. Натиснемо на кнопку **Next**.

І нам буде запропоновано встановити конфігурацію для другого продукту - **MySQL Router**:

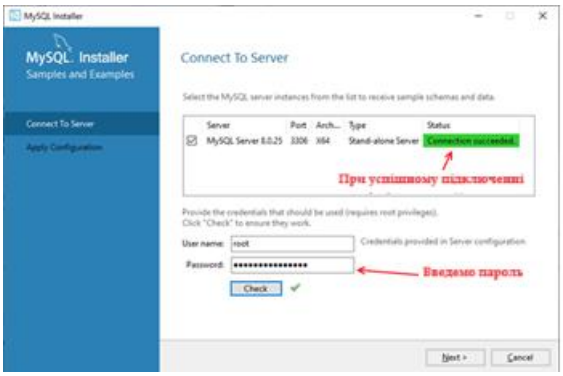


Нічого не будемо змінювати, залишивши усі налаштування за замовчуванням, і натиснемо на кнопку **Finish**.

Далі знову відобразиться вікно з переліком продуктів, готових до конфігурації. Натиснемо на кнопку **Next**.

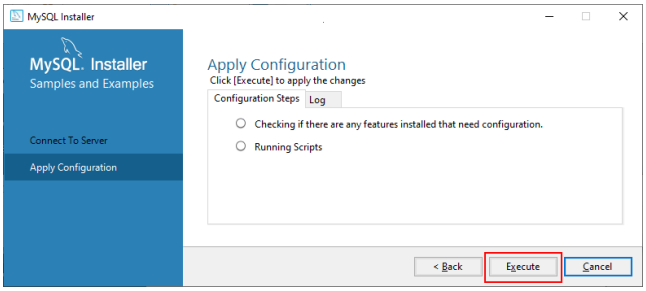


І далі нам буде запропоновано встановити конфігурацію для третього продукту - **Samples and Examples** (Приклади роботи з MySQL). Зокрема, треба буде вказати екземпляр сервера MySQL для отримання прикладів для роботи з MySQL. Встановлений екземпляр буде автоматично відмічено у списку. Крім того, пропонує протестувати підключення. У полі **Password** введемо раніше вказаний пароль і натиснемо на кнопку **Check**

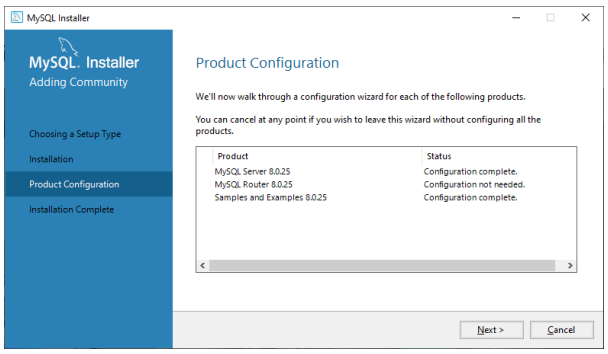


При успішному підключенні до MySQL відобразиться виділена зеленим кольором напис **Connection succeeded**. Натиснемо на кнопку Next.

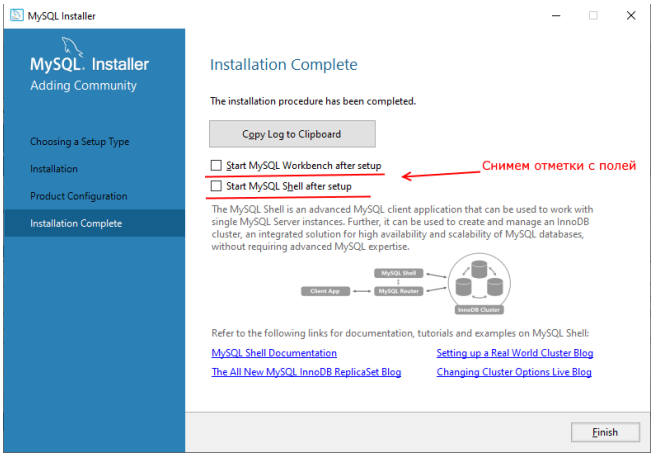
І на останньому вікні необхідно буде застосувати конфігурацію для



Далі ми знову побачимо вікно з переліком встановлених і сконфігурованих продуктів. І натиснемо на кнопку **Next.**



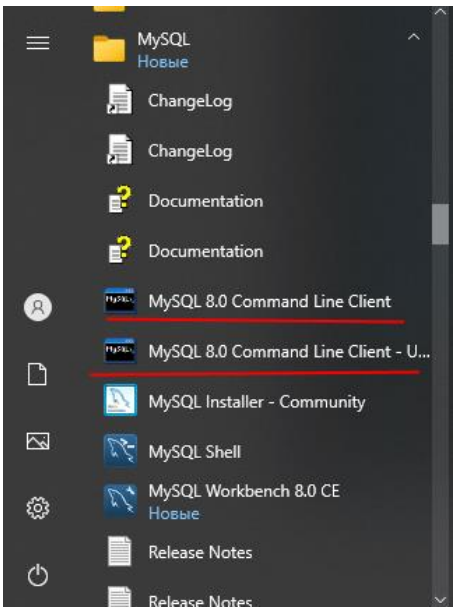
На останньому екрані ми побачимо два відмічені поля: **Start MySQL Workbench after setup** та **Start MySQL Shell after setup**. Ці поля дозволяють запустити графічний та консольний клієнти для управління сервером MySQL. Знімемо відмітки з цих полів, оскільки поки ми не збираємося запускати відповідні програми.



І натиснемо на кнопку Finish. Усе! MySQL повністю встановлено, сконфігуровано і запущено. І ми зможемо з ним працювати.

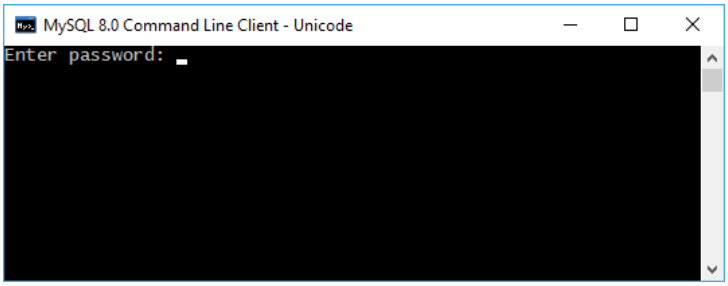
Консольний клієнт MySQL Command Line Client

При встановленні сервера MySQL також встановлюється консольний клієнт для роботи з базами даних. Наприклад, у Windows у меню Пуск можна знайти програму **MySQL 8.0 Command Line Client**. Це і є власне консольний клієнт:



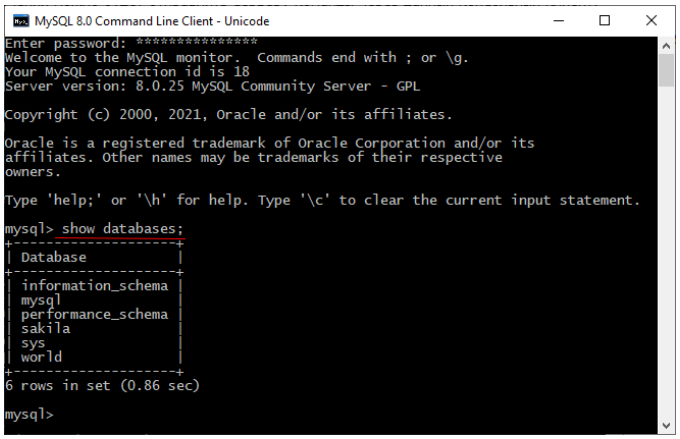
Причому клієнт встановлюється одразу у двох варіантах – з підтримкою Unicode та без.

Запустимо MySQL 8.0 Command Line Client - Unicode. Спочатку нам відобразиться запропоновано ввести пароль:



Тут необхідно ввести пароль, який був встановлений при встановленні MySQL для користувача root. І після успішного підключення можна буде відправляти серверу команди через даний консольний клієнт.

Для початку подивимося, які бази даних є на сервері за замовчуванням. Для цього введемо команду **show databases;**



Після виконання цієї команди ми побачимо, що на сервері за замовчуванням уже є низка баз даних, які виконують адміністративні функції.

Тепер створимо базу даних за допомогою наступної команди мови SQL:

create database test;

Для створення бази даних застосовується команда **create database**, після якої вказується назва БД. Тобто в даному випадку база даних називатиметься "test".

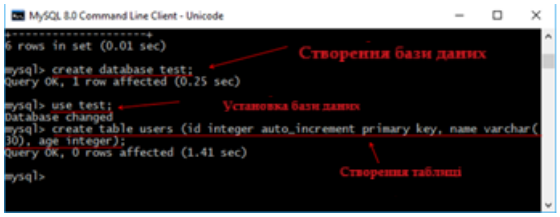
Щоб звертатися до якої-небудь певної бази даних, спочатку треба встановити потрібну базу даних як поточну. Для цього потрібно виконати команду **use**, після якої вказується назва бази даних. Наприклад, для підключення раніше створеної бази даних test введемо наступну команду:

use test;

Потім створимо в цій базі даних таблицю за допомогою команди:

create table users (id integer auto_increment primary key, name varchar(30), age integer);

Ця команда створює таблицю users, у якій буде три стовпці – id, name та age. id зберігатиме унікальний числовий ідентифікатор користувача і буде автоматично генеруватися базою даних, name зберігатиме ім'я користувача, а age – його вік.

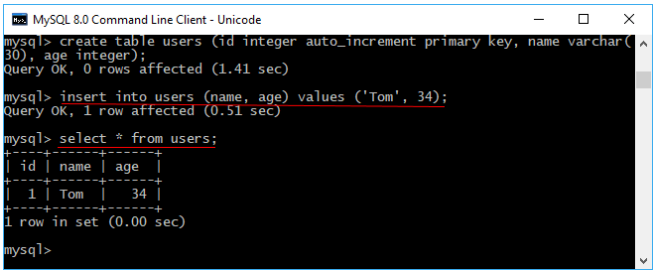


Після цього ми можемо додавати та отримувати дані з вище створеної таблиці. Спочатку додамо в таблицю один рядок за допомогою наступної команди:

insert into users (name, age) values ('Tom', 34);

І в кінці отримаємо додані дані:

select * from users;



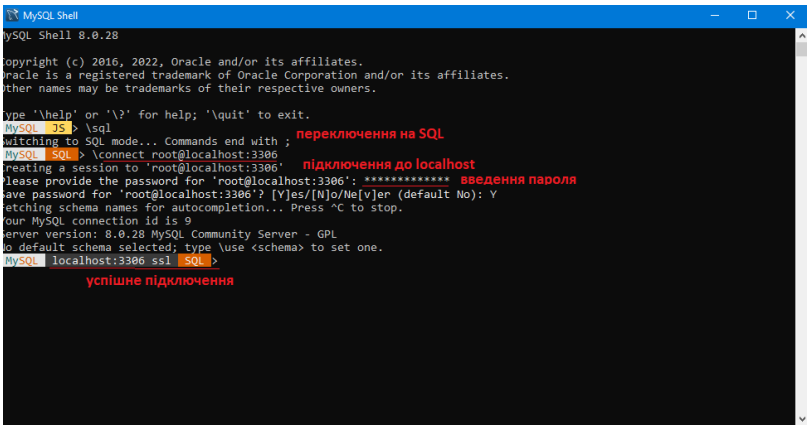
Таким чином, загалом ми можемо працювати з консольним клієнтом MySQL Command Line Client.

Консольний клієнт MySQL Shell

У минулій темі розглядався консольний клієнт **MySQL Command Line Client**, за допомогою якого можна керувати даними на сервері. Однак у останніх версіях MySQL також був доданий ще один консольний клієнт – **MySQL Shell**. Це більш сучасний, більш досконалий консольний клієнт, який надає трохи більше функціональності, ніж традиційний **MySQL Command Line Client**. Подивимося, як ми можемо з ним працювати.

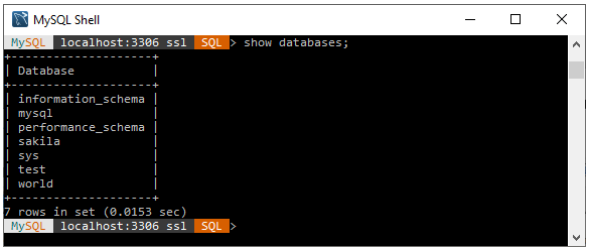
Якщо цільова ОС – Windows, то в меню Пуск у секції **MySQL** можна знайти програму **MySQL Shell**:

Запустимо цю програму. MySQL Shell підтримує введення команд трьома мовами: JavaScript, Python та SQL. Для встановлення мови, що використовується, застосовуються наступні команди: \js, \py та \sql. За замовчуванням застосовується JavaScript. Але оскільки ми будемо використовувати SQL, то переключимося на цю мову, ввівши команду **\sql**



Для взаємодії з сервером MySQL спочатку необхідно підключитися до нього. Для цього застосовується команда **\connect**, після якої вказується ідентифікатор (uri) у форматі ім'я_користувача@хост:порт. Оскільки в

більшості випадків використовують локальний сервер MySQL, який запущено на порту 3306, а для сервера mySQL доступний як мінімум один користувач – **root**, то можна використовувати для підключення наступний ідентифікатор: **root@localhost:3306**. Інакше треба підправити або ім'я користувача, або адресу, або порт.

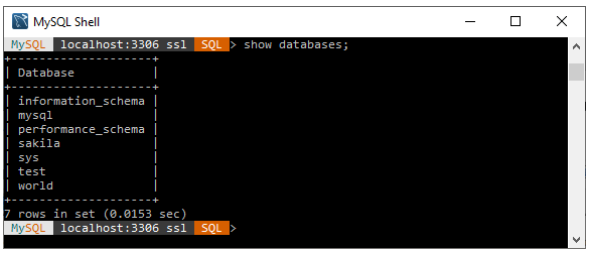


Після введення цієї команди програма запропонує ввести пароль для користувача root. І після успішного підключення можна буде відправляти серверу команди через MySQL Shell.

Для прикладу зробимо всі ті ж самі дії, що проводилися з **MySQL Command Line Client** у минулій темі.

Для початку подивимося, виведемо список баз даних, які є на сервері. Для цього введемо команду

show databases;



Тепер створимо базу даних за допомогою наступної команди мови SQL:

create database test2;

Для створення бази даних застосовується команда **create database**, після якої вказується назва БД. Тобто в даному випадку база даних називатиметься "test2".

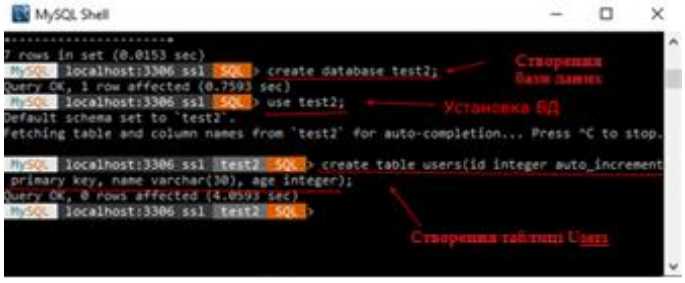
Щоб звертатися до якої-небудь певної бази даних, спочатку треба встановити потрібну базу даних як поточну. Для цього потрібно виконати команду **use**, після якої вказується назва бази даних. Наприклад, для підключення раніше створеної бази даних test введемо наступну команду:

use test2;

Потім створимо в цій базі даних таблицю за допомогою команди:

create table users (id integer auto_increment primary key, name varchar(30), age integer);

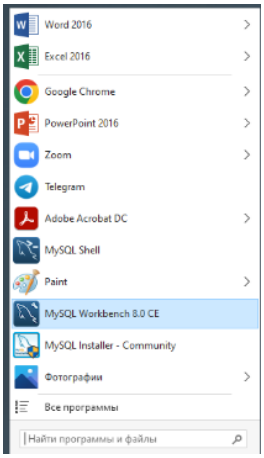
Ця команда створює таблицю users, у якій буде три стовпці – id, name та age. id зберігатиме унікальний числовий ідентифікатор користувача і буде автоматично генеруватися базою даних, name зберігатиме ім'я користувача, а age – його вік.



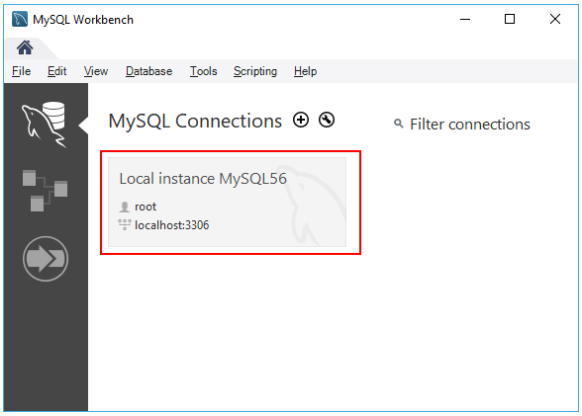
Графічний клієнт MySQL Workbench

Для спрощення роботи з сервером MySQL до базового комплекту встановлення входить такий інструмент як **MySQL Workbench**. Він представляє графічний клієнт для роботи з сервером, через який ми у зручному вигляді можемо створювати, видаляти, змінювати бази даних та керувати ними. Так, на Windows після встановлення в меню Пуск ми можемо знайти значок програми і запустити її:

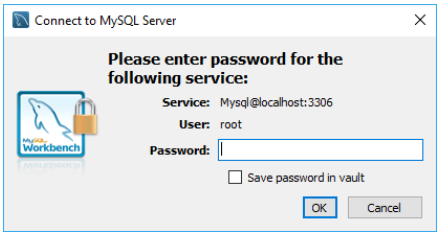
Для спрощення роботи з сервером MySQL до базового комплекту встановлення входить такий інструмент як MySQL Workbench. Він представляє графічний клієнт для роботи з сервером, через який ми у зручному вигляді можемо створювати, видаляти, змінювати бази даних та керувати ними. Так, на Windows після встановлення в меню Пуск ми можемо знайти значок програми і запустити її:



Нам відкриється наступне вікно, де ми можемо побачити поле з назвою запущеного локально екземпляра MySQL:

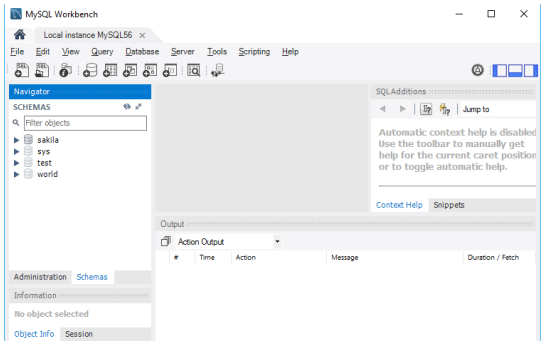


Натиснемо на нього, і нам відобразиться вікно для введення пароля:



Тут треба ввести пароль, який був встановлений для користувача root при встановленні MySQL.

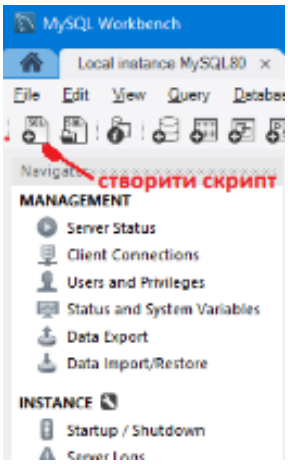
Після успішного логіну нам відкриється вміст сервера:



Зокрема, у лівій частині у вікні SCHEMAS можна побачити доступні бази даних.

2. *Створення бази даних*

Тепер подивимося, як ми можемо виконувати в цій програмі запити до БД. Спочатку створимо саму БД. Для цього натиснемо над списком баз даних на значок "SQL" з плюсом:

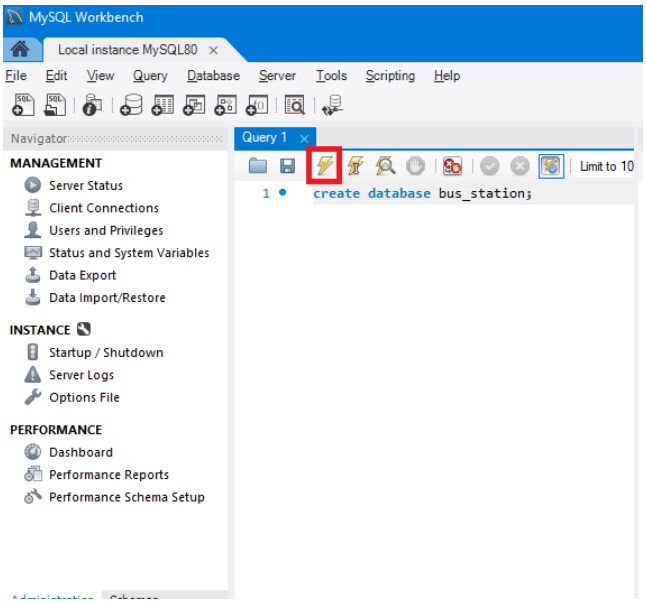


Після цього у центральній частині програми відкриється вікно для введення скрипту SQL. Введемо у нього наступну команду:

```
CREATE DATABASE usersdb;
```

Ця команда створює базу даних usersdb.

Для виконання скрипту на панелі інструментів натиснемо на значок блискавки:



Після цього внизу програми у полі виводу у разі успішного виконання ми побачимо зелений маркер та звіт про виконання.

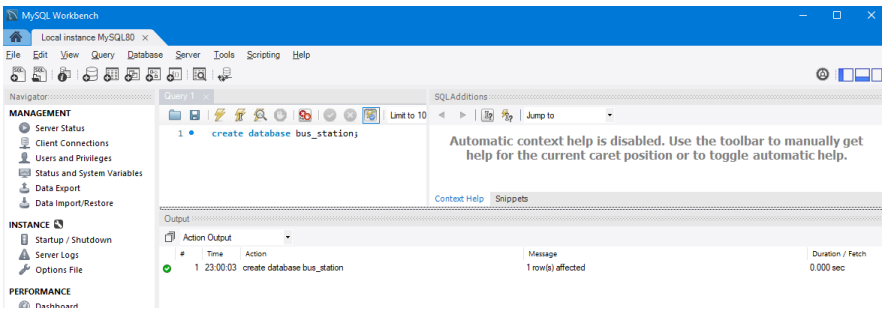
Таким чином, БД створено. Тепер додамо до неї таблицю та які-небудь дані. Для цього змінимо код у полі введення скрипту на наступний:

```
USE usersdb;
CREATE TABLE users (
   id INTEGER AUTO_INCREMENT PRIMARY KEY,
   firstname VARCHAR(30),
   age INTEGER
);
INSERT INTO users (firstname, age) VALUES ('Tom', 34);
```

Усі команди відокремлюються одна від одної крапкою з комою. Перша команда - USE встановлює як використовувану базу даних usersdb, яка була створена вище. Друга команда - CREATE TABLE створює в БД таблицю users, у якій буде три стовпці: id, firstname та age. Третя команда - INSERT INTO додає в таблицю users один рядок. Для виконання цих команд також натиснемо на значок блискавки.

І в кінці отримаємо всі дані з таблиці users за допомогою наступних команд:

```
USE usersdb;
SELECT * FROM users;
```



Таким чином, ми можемо здійснювати запити до БД у програмі MySQL Workbench CE.

3. Заповнення даними таблиць

Для заповнення таблиць даними користуються конструкцією SQL

INSERT [*LOW_PRIORITY*|*DELAYED*|*HIGH_PRIORITY*][IGNORE] **INTO** tablename (column1, column2, column3...) **VALUES** (value1, value2, value3, ...);

Тут значення параметрів наступні:

- **LOW_PRIORITY:** Цей модифікатор інформує MySQL про затримку виконання інструкції INSERT до того моменту, поки не буде ніяких зв'язків із зчитуванням з таблиці, яку ми намагаємось ВСТАВИТИ. Це допомагає досягти узгодженості всіх інших операцій, які будуть виконуватися.
- ***HIGH_PRIORITY*:** Цей модифікатор інформує MySQL про надання високого пріоритету інструкції INSERT перед будь-якою іншою заявою / транзакція що виконується.
- **IGNORE :** Цей модифікатор повідомляє MySQL Engine про ігнорування будь-яких помилок, які можуть виникнути внаслідок виконання оператора INSERT. Будь-які помилки, які виникають, трактуватимуться як прості попередження, а вставка записів у таблицю буде проходити безперешкодно.
- ***DELAYED*:** Це розширення MySQL до стандартного SQL. Коли користувач видає INSERT DELAYED, сервер ставить у чергу всі рядки, а дані вставляються в таблицю пізніше, коли таблиця не використовується іншими транзакціями.

Ця конструкція дозволяє заповнювати таблиці по записах.

Наприклад, створення таблиці та її заповнення виглядає так.

```
CREATE TABLE Educational_events (
    id INT PRIMARY KEY AUTO_INCREMENT,
    type_of_events VARCHAR(100),
    place VARCHAR(100),
    date DATE,
    id_shedule INT,
    id_finance INT,
    FOREIGN KEY (id_shedule) REFERENCES Scedule(id),
    FOREIGN KEY (id_finance) REFERENCES Finance(id)
);
```

```
INSERT INTO Educational_events (type_of_events, place, date, id_shedule, id_finance)
VALUES
    ("Лекція", "Львів", "2024-04-01", 1, 3),
    ("Вишкіл", "Київ", "2024-04-05", 2, 4),
    ("Похід", "Карпати", "2024-04-10", 3, 5),
    ("Таборування", "Полтавська обл.", "2024-04-15", 4, 6),
    ("Спортивні змагання", "Івано-Франківськ", "2024-04-20", 5, 7),
    ("Вишкіл з медицини", "Чернівці", "2024-04-25", 6, 8),
    ("Екскурсія", "Карпатський національний парк", "2024-05-01", 7, 9),
    ("Основи виживання", "Київ", "2024-05-05", 8, 10),
    ("Вогняні церемонії", "Львів", "2024-05-10", 9, 11),
    ("Лідерський тренінг", "Вінниця", "2024-05-15", 10, 12),
    ("Табір", "Закарпаття", "2024-05-20", 11, 13),
    ("Орієнтування", "Черкаси", "2024-05-25", 12, 14),
    ("Фестиваль пластової пісні", "Київ", "2024-06-01", 13, 15),
    ("Вишкіл з альпінізму", "Карпати", "2024-06-05", 14, 16),
    ("Екологічна акція", "Львів", "2024-06-10", 15, 17),
    ("Практикум з картографії", "Одеса", "2024-06-15", 16, 18),
    ("Вишкіл з медицини", "Полтава", "2024-06-20", 17, 19),
```

Лабораторна робота №6

Розробка запитів до бази даних у mySQL

ЗАВДАННЯ

Реалізувати запити такої складності.

1.Запит, джерелом інформації в якій є одна таблиця, а результуючим набором не всі її поля.

2. Запит по одній таблиці із використанням простих умов.

3. Запит по одній таблиці із використанням складених умов.

4. Запити по декільком зв`язаним таблицям.

5.Запит по декільком зв`язаним таблицям з використанням простих чи складених умов.

6. Запит по одній або декількох зв`язаних таблиць з використанням сортування.

7. Запит по одній або декількох зв`язаних таблиць з використанням обчислювальних полів.

8. Запит по одній або декількох зв`язаних таблиць з використанням агрегатних функцій.

9. Запит по одній або декількох зв`язаних таблиць з використанням групування.

10. Запит по одній або декількох зв`язаних таблиць з використанням групування та умови, що накладаються на групи.

11. Запит по одній або декількох зв`язаних таблиць з використанням групування та сортуванням.

12. Запит по одній або декількох зв`язаних таблиць з використанням групування та умови, що накладаються на групи та сортуванням.

13. Запит по одній або декількох зв`язаних таблиць з використанням простого підзапиту.

14. Запит по одній або декількох зв`язаних таблиць з використанням корельованого підзапиту.

МЕТОДИЧНІ ВКАЗІВКИ

Для реалізації запитів довільної складності використовують конструкцію SQL

SELECT список полів

FROM список таблиць

WHERE умови відбору записів

GROUP BY список полів

HAVING умови відбору груп

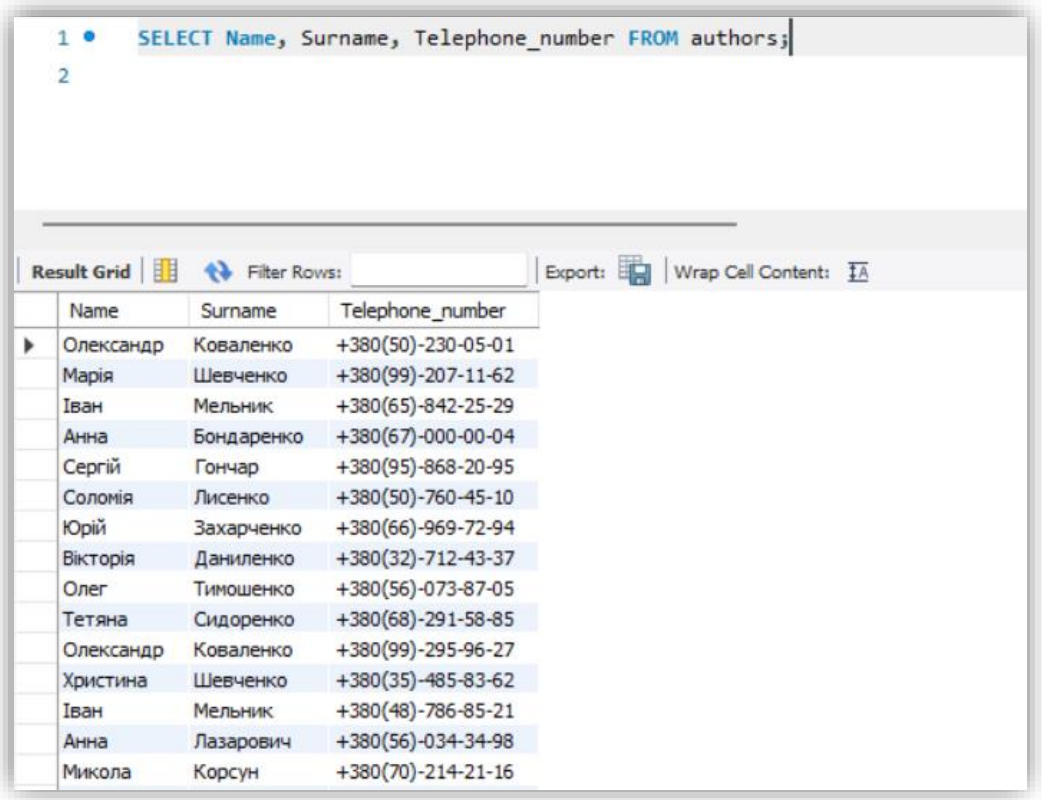
ORDER BY список полів1;

Наведемо приклади реалізації запитів різної складності.

1.Запит, джерелом інформації в якій є одна таблиця, а результуючим набором не всі її поля.

Скласти список авторі, вказавши їх номери телефонів

SELECT Name, Surname, Telephone_number FROM authors;



```
1 • SELECT Name, Surname, Telephone_number FROM authors;
2
```

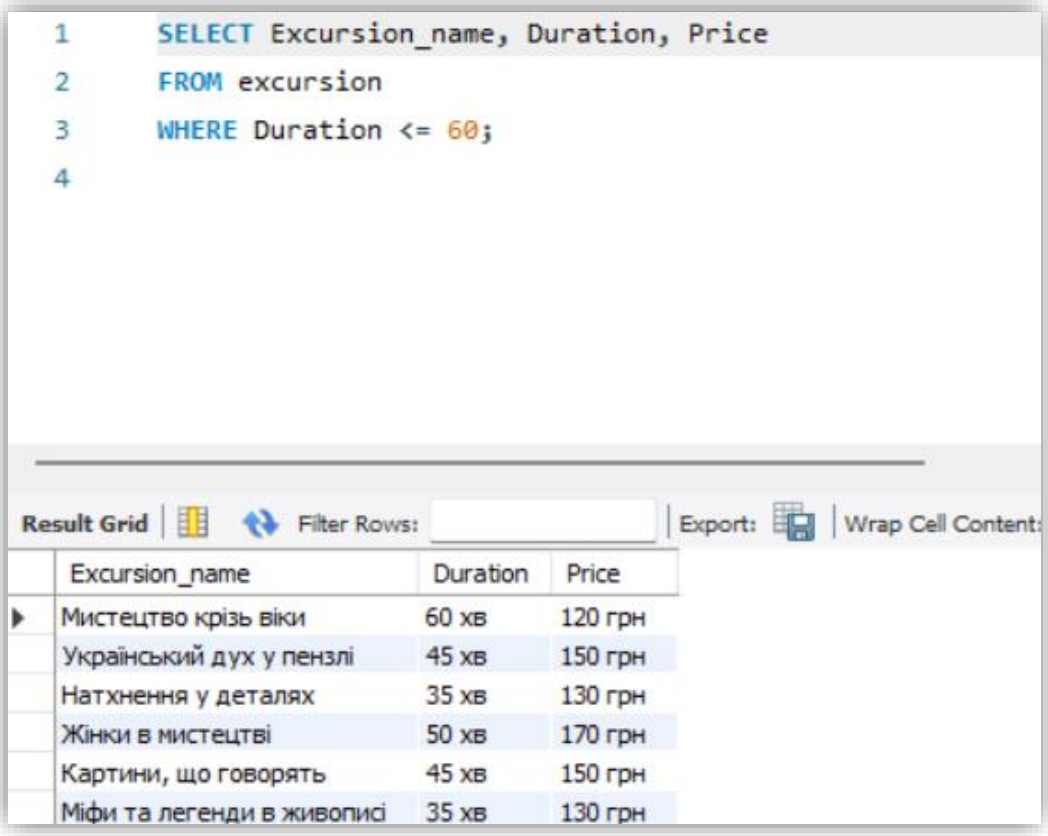
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Name	Surname	Telephone_number
Олександр	Коваленко	+380(50)-230-05-01
Марія	Шевченко	+380(99)-207-11-62
Іван	Мельник	+380(65)-842-25-29
Анна	Бондаренко	+380(67)-000-00-04
Сергій	Гончар	+380(95)-868-20-95
Соломія	Лисенко	+380(50)-760-45-10
Юрій	Захарченко	+380(66)-969-72-94
Вікторія	Даниленко	+380(32)-712-43-37
Олег	Тимошенко	+380(56)-073-87-05
Тетяна	Сидоренко	+380(68)-291-58-85
Олександр	Коваленко	+380(99)-295-96-27
Христина	Шевченко	+380(35)-485-83-62
Іван	Мельник	+380(48)-786-85-21
Анна	Лазарович	+380(56)-034-34-98
Микола	Корсун	+380(70)-214-21-16

2. Запит по одній таблиці із використанням простих умов.

Скласти список екскурсій, їх цін та із обмеженням тривалості до години:

SELECT Excursion_name, Duration, Price
FROM excursion
WHERE Duration <= 60;



3. Запит по одній таблиці із використанням складених умов.

Вказати працівників на посаді екскурсовод, що мають ступінь освіти бакалавр і продавців-касирів, що мають вищу освіту:

```
SELECT Job_position, Name, Surname, Education
FROM workers
WHERE (Job_position = 'Екскурсовод' AND Education LIKE '%Бакалавр%')
   OR (Job_position = 'Продавець-касир' AND Education LIKE '%Вища%');
```

```sql
SELECT Job_position, Name, Surname, Education
FROM workers
WHERE (Job_position = 'Екскурсовод' AND Education LIKE '%Бакалавр%')
   OR (Job_position = 'Продавець-касир' AND Education LIKE '%Вища%');
```

esult Grid | Filter Rows: | Export: | Wrap Cell Content:

Job_position	Name	Surname	Education
Екскурсовод	Дмитро	Гончар	Бакалавр, факультет культурології
Екскурсовод	Тетяна	Мороз	Бакалавр, факультет історії
Продавець-касир	Світлана	Косар	Вища, факультет менеджменту культури та...
Продавець-касир	Ярослав	Морозов	Вища, факультет культурології
Екскурсовод	Гліб	Бондар	Бакалавр, факультет історії
Продавець-касир	Інна	Савченко	Вища, факультет історії
Екскурсовод	Платон	Лисенко	Бакалавр, факультет етнології

4. Запити по декільком зв`язаним таблицям.

Скласти перелік експонатів, вказавши поверх розташування, ім`я автора.

SELECT exhibit.Art_type, exhibit.Name, halls.Floor, authors.Name

FROM exhibit

JOIN halls ON exhibit.Hall_id = halls.id_hall

JOIN authors ON exhibit.Author_id = id_authors;

```sql
SELECT exhibit.Art_type, exhibit.Name, halls.Floor, authors.Name
FROM exhibit
JOIN halls ON exhibit.Hall_id = halls.id_hall
JOIN authors ON exhibit.Author_id = id_authors;
```

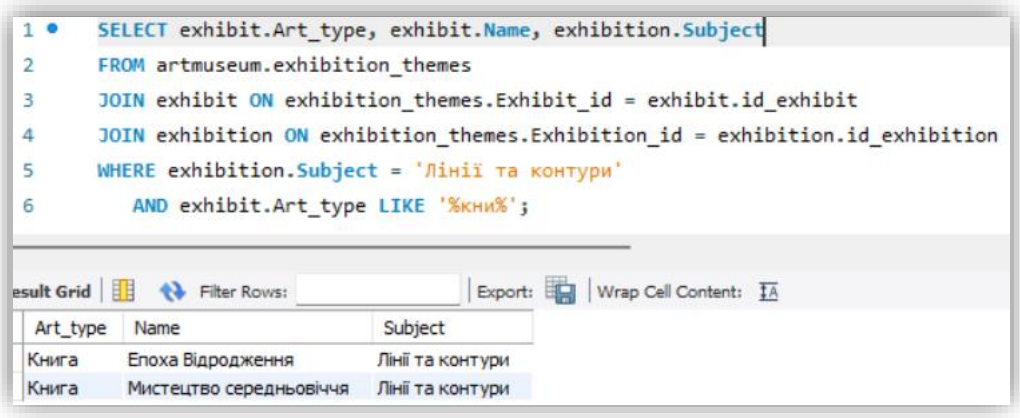
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Art_type	Name	Floor	Name
Ескізи	Ескізи для театральної сцени	1	Юрій
Ескізи	Ескізи до статуї	1	Соломія
Книга	Архітектура та стиль	1	Тетяна
Скульптура	Світло і тінь	1	Сергій
Ескізи	Ескізи до сучасного мистецтва	1	Вікторія
Ескізи	Ескізи до скульптури	1	Вікторія
Картина	Зимовий пейзаж	1	Вікторія
Ескізи	Сучасні тенденції	1	Микола
Книга	Історія мистецтва	1	Олег
Картина	Пейзаж весняний	2	Христина
Картина	Натюрморт	2	Юрій
Книга	Мистецтво на перехресті	2	Олександр
Ескізи	Ескізи для виставки	2	Тетяна
Картина	Нічне місто	2	Анна
Скульптура	Тема внутрішньої сили	2	Світлана

5. Запит по декільком зв`язаним таблицям з використанням простих чи складених умов.

Скласти перелік книг- експонатів, які будуть на виставці з темою «Лінії та контури».

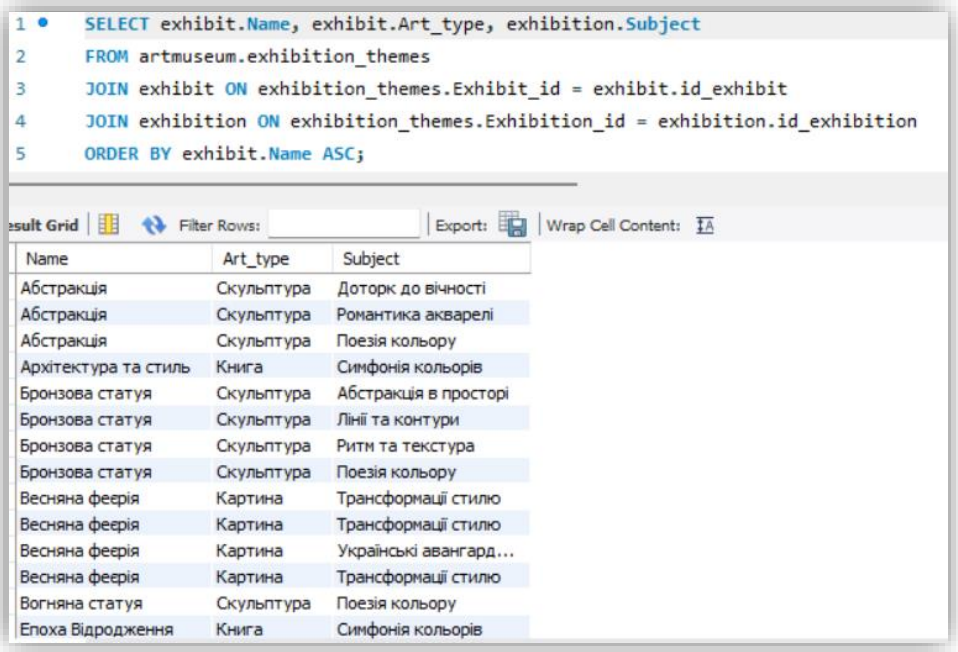
```
SELECT exhibit.Art_type, exhibit.Name, exhibition.Subject
FROM artmuseum.exhibition_themes
JOIN exhibit ON exhibition_themes.Exhibit_id = exhibit.id_exhibit
JOIN exhibition ON exhibition_themes.Exhibition_id = exhibition.id_exhibition
WHERE exhibition.Subject = 'Лінії та контури'
  AND exhibit.Art_type LIKE '%кни%';
```



6. Запит по одній або декількох зв`язаних таблиць з використанням сортування.

Вивести назви експонатів (в алфавітному порядку), їх тип та теми виставок, в яких вони беруть участь:

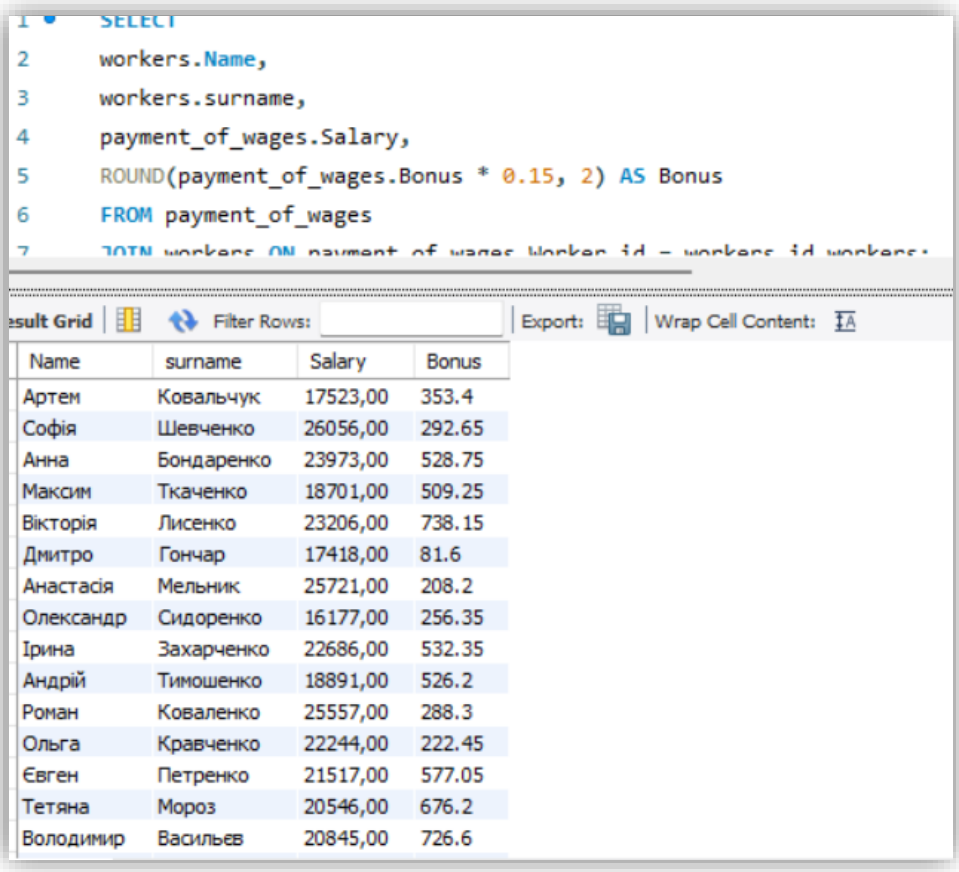
```
SELECT exhibit.Name, exhibit.Art_type, exhibition.Subject
FROM artmuseum.exhibition_themes
JOIN exhibit ON exhibition_themes.Exhibit_id = exhibit.id_exhibit
JOIN exhibition ON exhibition_themes.Exhibition_id = exhibition.id_exhibition
ORDER BY exhibit.Name ASC;
```



7. Запит по одній або декількох зв`язаних таблиць з використанням обчислювальних полів.

Запит на збільшення працівникам бонусу в 1,5 рази:

```
SELECT
workers.Name,
workers.surname,
payment_of_wages.Salary,
ROUND(payment_of_wages.Bonus * 0.15, 2) AS Bonus
FROM payment_of_wages
JOIN workers ON payment_of_wages.Worker_id = workers.id_workers;
```



8. Запит по одній або декількох зв’язаних таблиць з використанням агрегатних функцій.

Вивести працівників, що працюють в крамниці на касі, що продавали роботи Автора «63» і на яку суму вони продали

```
SELECT
    COUNT(*) AS total_operations,
    SUM(store.Amount_earned) AS total_amount
FROM store
WHERE store.Author_id = '63';
```

```
1 •  SELECT
2        COUNT(*) AS total_operations,
3        SUM(store.Amount_earned) AS total_amount
4    FROM store
5    WHERE store.Author_id = '63';
```

esult Grid | Filter Rows: | Export: | Wrap Cell Content:

total_operations	total_amount
2	672

9. Запит по одній або декількох зв’язаних таблиць з використанням групування.

Для кожного реставратора порахувати кількість виконаних робіт.

```
SELECT workers.Name, workers.Surname,
    COUNT(restoration.Restorer_id) AS total_restorations
FROM restoration
JOIN workers ON restoration.Restorer_id = workers.id_workers
GROUP BY workers.id_workers, workers.Name, workers.Surname;
```

```
1 •  SELECT workers.Name, workers.Surname,
2    COUNT(restoration.Restorer_id) AS total_restorations
3    FROM restoration
4    JOIN workers ON restoration.Restorer_id = workers.id_workers
5    GROUP BY workers.id_workers, workers.Name, workers.Surname;
6
```

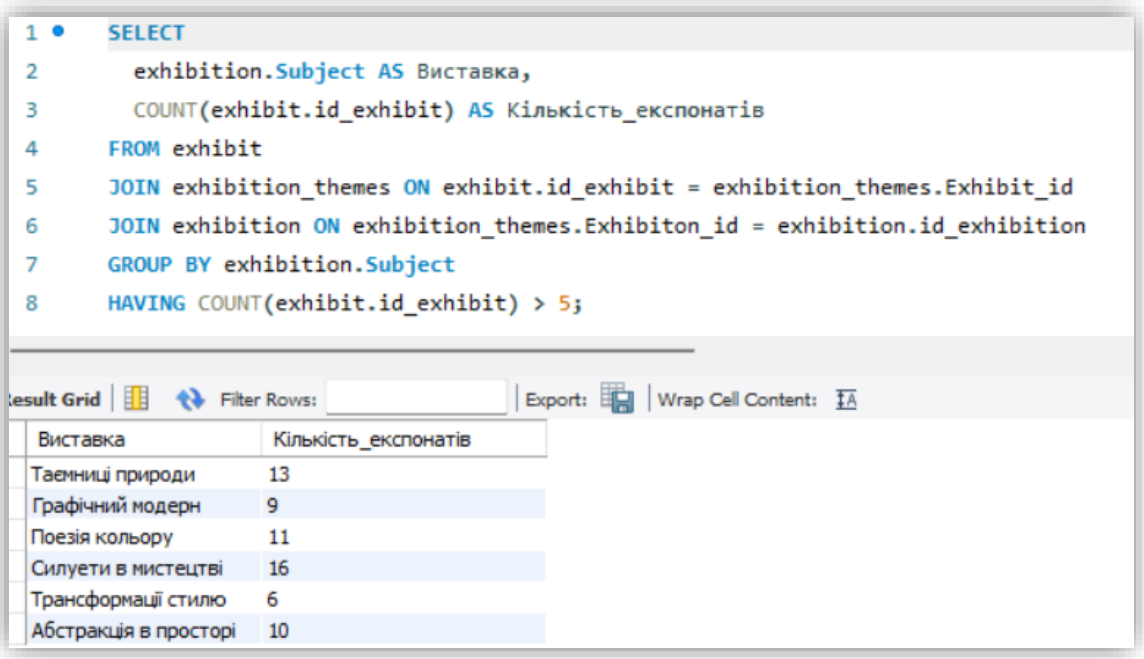
sult Grid | Filter Rows: | Export: | Wrap Cell Content:

Name	Surname	total_restorations
Ірина	Захарченко	10
Христина	Поліщук	12
Тетяна	Барбара	9
Тарас	Левченко	9
Антон	Петров	15
Роман	Силан	13
Єгор	Бондаренко	11
Аліна	Кузьменко	9
Іван	Коваленко	12

10. Запит по одній або декількох зв’язаних таблиць з використанням групування та умови, що накладаються на групи.

Скласти перелік виставок, у яких кількість експонатів перевищує 5.

```
SELECT
  exhibition.Subject AS Виставка,
  COUNT(exhibit.id_exhibit) AS Кількість_експонатів
FROM exhibit
JOIN exhibition_themes ON exhibit.id_exhibit = exhibition_themes.Exhibit_id
JOIN exhibition ON exhibition_themes.Exhibiton_id = exhibition.id_exhibition
GROUP BY exhibition.Subject
HAVING COUNT(exhibit.id_exhibit) > 5;
```

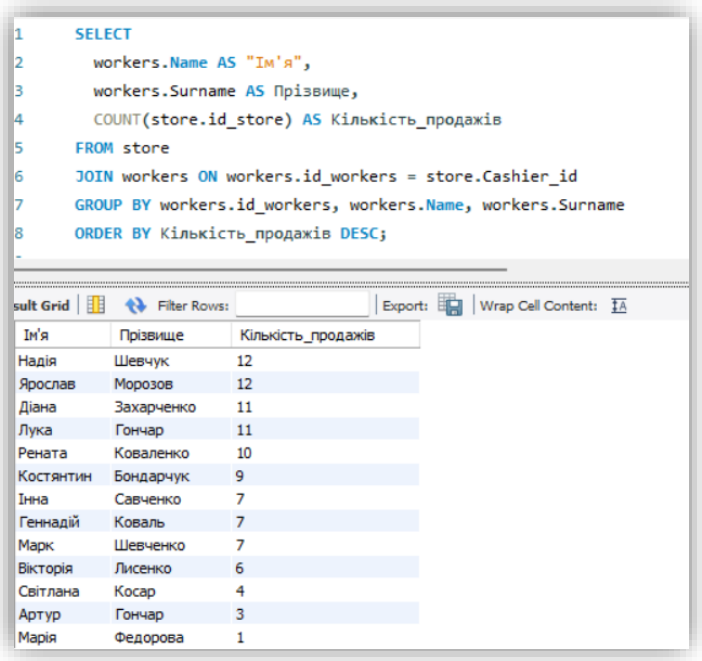


11. Запит по одній або декількох зв’язаних таблиць з використанням групування та сортуванням.

Вивести касирів та кількість їх продажів:

```
SELECT
  workers.Name AS "Ім'я",
```

```
  workers.Surname AS Прізвище,
  COUNT(store.id_store) AS Кількість_продажів
FROM store
JOIN workers ON workers.id_workers = store.Cashier_id
GROUP BY workers.id_workers, workers.Name, workers.Surname
ORDER BY Кількість_продажів DESC;
```



```
SELECT
  workers.Name AS "Ім'я",
  workers.Surname AS Прізвище,
  COUNT(store.id_store) AS Кількість_продажів
FROM store
JOIN workers ON workers.id_workers = store.Cashier_id
GROUP BY workers.id_workers, workers.Name, workers.Surname
ORDER BY Кількість_продажів DESC;
```

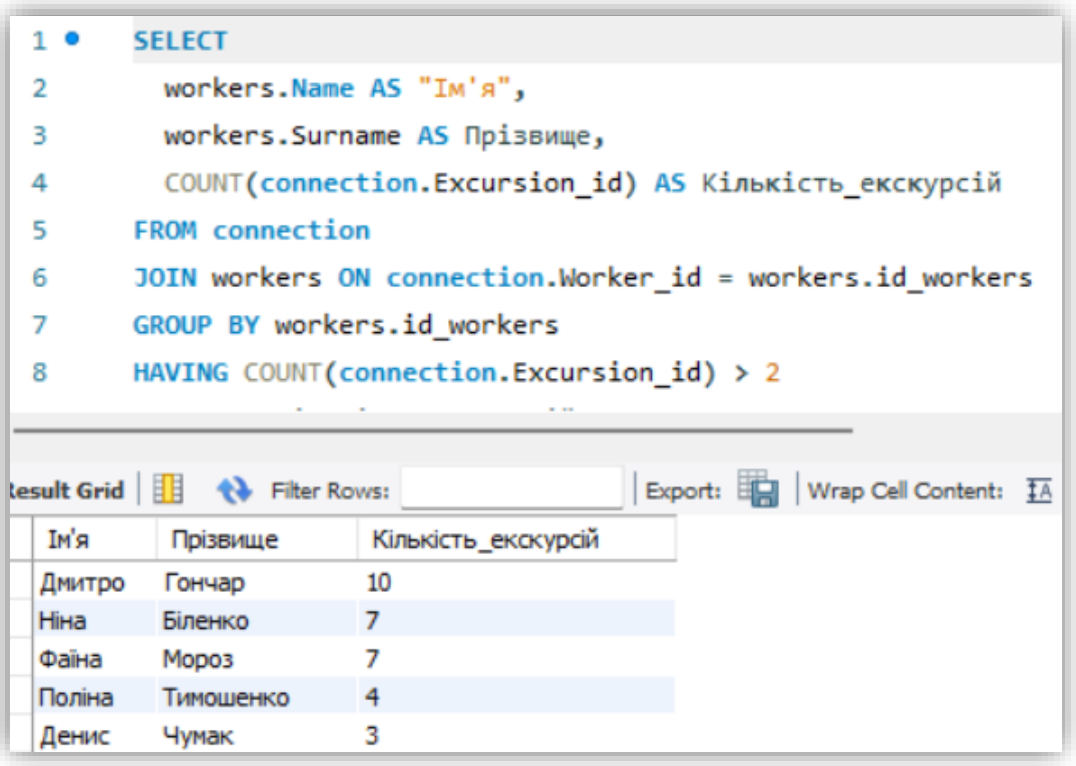
Filter Rows: | Export: | Wrap Cell Content:

Ім'я	Прізвище	Кількість_продажів
Надія	Шевчук	12
Ярослав	Морозов	12
Діана	Захарченко	11
Лука	Гончар	11
Рената	Коваленко	10
Костянтин	Бондарчук	9
Інна	Савченко	7
Геннадій	Коваль	7
Марк	Шевченко	7
Вікторія	Лисенко	6
Світлана	Косар	4
Артур	Гончар	3
Марія	Федорова	1

12. Запит по одній або декількох зв’язаних таблиць з використанням групування та умови, що накладаються на групи та сортуванням.

Вивести екскурсоводів, що проводять більше 2 екскурсій:

```
SELECT
  workers.Name AS "Ім'я",
  workers.Surname AS Прізвище,
  COUNT(connection.Excursion_id) AS Кількість_екскурсій
FROM connection
JOIN workers ON connection.Worker_id = workers.id_workers
GROUP BY workers.id_workers
HAVING COUNT(connection.Excursion_id) > 2
ORDER BY Кількість_екскурсій DESC;
```



```
SELECT
  workers.Name AS "Ім'я",
  workers.Surname AS Прізвище,
  COUNT(connection.Excursion_id) AS Кількість_екскурсій
FROM connection
JOIN workers ON connection.Worker_id = workers.id_workers
GROUP BY workers.id_workers
HAVING COUNT(connection.Excursion_id) > 2
```

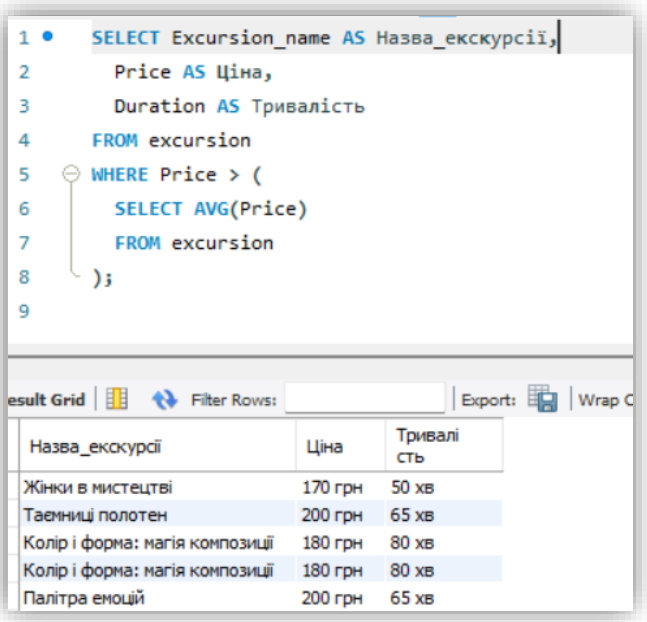
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Ім'я	Прізвище	Кількість_екскурсій
Дмитро	Гончар	10
Ніна	Біленко	7
Фаїна	Мороз	7
Поліна	Тимошенко	4
Денис	Чумак	3

13. Запит по одній або декількох зв’язаних таблиць з використанням простого підзапиту.

Відобразити екскурсії, чия тривалість вища за середню:

```
SELECT Excursion_name AS Назва_екскурсії,
  Price AS Ціна,
  Duration AS Тривалість
FROM excursion
WHERE Price > (
  SELECT AVG(Price)
  FROM excursion
);
```



14. Запит по одній або декількох зв’язаних таблиць з використанням корельованого підзапиту.

Скласти таблицю картин, які беруть участь хоча б в одній виставці:

```
SELECT Name AS Назва,
Hall_number AS Номер_залу
FROM exhibit
JOIN halls ON exhibit.Hall_id = halls.id_hall
WHERE EXISTS (
  SELECT 1
  FROM exhibition_themes
  WHERE exhibition_themes.Exhibit_id = exhibit.id_exhibit and Art_type = 'Картина'
);
```

```sql
SELECT Name AS Назва,
Hall_number AS Номер_залу
FROM exhibit
JOIN halls ON exhibit.Hall_id = halls.id_hall
WHERE EXISTS (
  SELECT 1
  FROM exhibition_themes
  WHERE exhibition_themes.Exhibit_id = exhibit.id_exhibit and Art_type = 'Картина'
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Назва	Номер_залу
Осінь в лісі	17
Нічне місто	8
Весняна феерія	18
Літній пейзаж	15
Погляд у далечінь	12
Натюрморт	5
Зимовий пейзаж	3
Сонце і місяць	8

Список літератури

1. C.J.Date An Instruction to Database System. – 2008. – 1024 с. Електронна книга.
2. А. Ю. Берко, О. М. Верес, В. В. Пасічник Системи баз даних та знань. Книга 2. Системи управління базами даних та знань: навч. посібник. – Львів: «Магнолія-2006».–2020.– 584 с.
3. .Павлишин, Л.Гліненко, Н. Шаховська Основи інформаційних технологій і систем. – Л.: Львівська політехніка, 2018.- 620 с.
4. Трофименко О. Г. Організація баз даних : навч. посібник / О. Г. Трофименко, Ю. В. Прокоп, Н. І. Логінова, І. М. Копитчук. 2-ге вид. виправ. і доповн. – Одеса : Фенікс, 2019. – 246 с.

Додаток 1. Теми предметних областей

1. Поліклініка.
2. Хлібокомбінат.
3. Міські пасажирські перевезення.
4. Підприємство з виготовлення молочної продукції
5. Лікарня.
6. Курси іноземних мов
7. Військова частина.
8. Будівельна організація.
9. ЗНО випускників ЗОШ.
10. Спортивна організація.
11. Музей
12. Гуртки дозвілля для учнів ЗОШ.
13. Інформаційна система для готелю.
14. Інформаційна система продовольчого магазину.
15. Драматичний театр.
16. Туристична фірма.
17. Аптека.
18. Телефонні розмови мобільного зв`язку.
19. Аеропорт
20. Ветеринарна лікарня.
21. Зоопарк
22. Міська філармонія
23. Залізничний вокзал
24. Фітнес клуб
25. Станція технічного обслуговування (мережа СТО)
26. Тваринна ферма.
27. Автовокзал
28. Інформаційна модель меблевого магазину.
29. Страхові компанії.
30. Ізолятор тимчасового утримання громадян-порушників.
31. Інформаційна система водних ресурсів країни.
32. Інформаційна система автомобільних доріг.
33. Бібліотека.
34. Інформаційна система провайдера (інтернет послуги).
35. Інформаційна система радіостанції
36. Інформаційна система «Ресторан»
37. Інформаційна система «Молодіжна організація «Пласт»»
38. Інформаційна система «Магазин будівельних матеріалів»
39. Інформаційна система «Розважальний центр для дітей»
40. Інформаційна система «Лісового господарства»
41. Архів документів
42. Інформація про ВПО.
43. Інформація для пошуку зниклих
44. Інформаційна система «Пошта».

Навчальне видання

Методичні вказівки та завдання до лабораторних робіт

Укладачі: *Іліка Світлана Анатоліївна*,

Піддубна Лариса Андріївна,

Відповідальний за випуск *Черевко І.М.*.