МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ТЕРНОПІЛЬСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ КОМП'ЮТЕРНИХ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Л.М. Буяк, А.Я. Мушак, Н.Г. Хома

ПРАЦЮЄМО З БАЗАМИ ДАНИХ В СЕРЕДОВИЩІ Microsoft Office: теоретичні аспекти та приклади розв'язування задач

Навчальний посібник з курсу „Сучасні інформаційні технології" для студентів денної та заочно-дистанційної форм навчання галузей знань 07 „Управління та адміністрування", 23 „Соціальна робота"
ступеня вищої освіти „бакалавр"

У цьому друкованому виданні містяться теоретичні відомості для роботи з базами даних і подані вказівки щодо розв'язання найтиповіших завдань з найактуальніших напрямків економічної думки

Тернопіль ТНЕУ
2019

Буяк Л.М., Мушак А.Я., Хома Н.Г. **Працюємо з базами даних в середовищі Microsoft Office: теоретичні аспекти та приклади розв'язування задач** Навчальний посібник з курсу „Сучасні інформаційні технології" для студентів денної та заочно-дистанційної форм навчання галузей знань 07 „Управління та адміністрування", 23 „Соціальна робота" ступеня вищої освіти „бакалавр". – Тернопіль: ТНЕУ, 2019. – 80 с.

Навчальний посібник містить теоретичний матеріал, який стосується основ роботи з базами даних в середовищі Microsoft Office та приклади розв'язування задач створення та підтримки типових баз даних. Структура подання теоретичного матеріалу така: 1 - Основи теорії баз даних. 2. Робота з однотабличними базами даних в середовищі Excel. 3. Робота з реляційними базами даних в середовищі Access. Наведені приклади розв'язування п'яти задач, перші дві з яких та остання демонструють роботу в середовищі СУБД Access, а решта - в середовищі табличного процесора Excel в контексті баз даних робочого аркуша.

Автори: ***Буяк Леся Михайлівна***,
доктор економічних наук,
професор, завідувач кафедри економічної кібернетики та інформатики
Мушак Андрій Ярославович,
кандидат технічних наук,
доцент кафедри
економічної кібернетики та інформатики
Хома Надія Григорівна,
кандидат фізико-математичних наук,
доцент кафедри
економічної кібернетики та інформатики

Рецензенти: ***Кнейслер Ольга Володимирівна***,
доктор економічних наук, професор, завідувач кафедри фінансового менеджменту та страхування Тернопільського національного економічного університету

Балик Надія Романівна,
кандидат педагогічних наук,
доцент, завідувач кафедри інформатики та методики її навчання Тернопільського національного педагогічного університету імені Володимира Гнатюка

Відповідальний за випуск: ***Буяк Леся Михайлівна***,
доктор економічних наук,
доцент, завідувач кафедри
економічної кібернетики та інформатики

Затверджено
на засіданні кафедри економічної кібернетики та інформатики,
протокол № 3 від 11 жовтня 2019 р.

Розглянуто та схвалено
вченою радою факультету комп'ютерних інформаційних технологій,
протокол № 2 від 31 жовтня 2019 р.

© Л.М. Буяк, А.Я. Мушак, Н.Г. Хома

ВСТУП

Програмний пакет Microsoft Office містить поряд із класичним засобом для створення і підтримки функціонування баз даних – СУБД Access, – ще й систему для роботи зі списками (базами даних робочого аркуша), яка функціонує в рамках табличного процесора Excel. Access є потужною системою управління реляційними базами даних, яка спрощує зберігання та доступ до великих обсягів даних. Коли потрібно постійно додавати нові дані або реорганізувати результати, не реорганізуючи при цьому основне джерело даних, то саме час скористатися допомогою Access. Excel може бути швидким і простим рішенням, коли просто потрібно відсортувати, відфільтрувати або обчислити відносно обмежену[1] кількість даних. Наступний виклад теоретичного матеріалу має таку послідовність:

1. Основи теорії баз даних.
2. Робота з однотабличними базами даних в середовищі Excel.
3. Робота з реляційними базами даних в середовищі Access.

[1] Таким лімітом є кількість рядків робочого аркуша Excel, про що детальніше йтиме мова в розділі „Технологія роботи з базами даних в середовищі електронних таблиць”.

РОЗДІЛ 1. ОСНОВИ ТЕОРІЇ БАЗ ДАНИХ

Основні компоненти баз даних

Словосполучення **база даних** має багато визначень, але в даному посібнику під базою даних (БД, **DB – DataBases**) будемо розуміти упорядкований, структурований набір даних, наприклад адреси клієнтів або назви товарів. При цьому слід пам'ятати що управління базою даних (впорядкування, сортування, пошук і т.ін.) можливо здійснювати лише за допомогою спеціальної програми обслуговування БД – **системи управління базою даних (СУБД)**. Тобто необхідно розуміти під базою даних набір даних разом із програмою їх обслуговування. Прикладом такої бази даних може бути електронний телефонний довідник, що містить номери телефонів і адреси абонентів та програму для їх перегляду.

Для уніфікації механізмів пошуку даних набір даних необхідно зберігати в стандартній табличній формі, де кожний **запис бази даних** є окремим рядком таблиці. Такий набір даних складається із записів, а кожний запис – з окремих **полів.** Структура всіх записів в кожній базі даних однакова і всі вони мають одну і ту ж послідовність полів, але вміст кожного поля інший.

Вимоги до баз даних

Спочатку бази даних створювали для розв'язку якої-небудь конкретної спеціалізованої задачі, як наприклад: список замовлень, журнал реєстрації студентів університету. Але з часом до проектування і створення бази даних розробили наступні вимоги:

- Необхідно мати єдине та постійне джерело даних, керування і доступ до якого могли б здійснювати одночасно різні програми.
- Для БД необхідний надійний механізм збереження і підтримки цілісності даних тому, що складні програми здатні одночасно використовувати дуже велику кількість інформації протягом дуже тривалого часу.
- Необхідно розмежовувати рівні доступу до даних. Звичайно база даних використовується невеликою групою користувачів: відділом підприємства чи робочою групою. Але часто база даних може виявитися необхідною, наприклад, цілій організації; у цьому випадку мова йде про базу даних масштабу підприємства. Можна створити базу даних, інформація в якій буде доступна тільки одній людині, незважаючи на те що дані (наприклад, платіжна відомість) були б цікаві будь-якому співробітнику підприємства.

Типи баз даних

Замість терміну ***база даних*** часто вживають термін **сховище даних** (**Data Warehouse**). Власне кажучи, це є просто склад інформації (як, наприклад, міський архів), що може бути використаним більш дрібними прикладними базами даних. При використанні Internet-ресурсів інформація зі сховищ даних найчастіше виступає основою того, що користувач бачить у своєму Web-браузері.

Дещо рідше використовується термін **магазин даних** (**Data Mart).** Це окремий різновид сховища даних для об'єднання баз даних відділу чи робочої групи без створення бази даних масштабу підприємства.

Дуже великими **VLDB** (**Very Large Data Bases**) називають бази даних, розмір яких досяг значної величини внаслідок:

– занадто тривалого збереження даних (архів міськвиконкому);
– збереження дуже об'ємних даних (малюнки, відео);
– високої частоти внесення даних (реєстрація дзвінків).

Дуже великі бази даних часто використовують разом зі сховищами даних. Сховище даних у таких випадках забезпечує потреби інших, менших баз даних, здійснюючи як збереження, так і обробку перерахованих вище типів даних, чи ж виступає у ролі центрального розподільника інформації для всіх прикладних програм. VLDB – це просто велика база даних і критерії для її визначення, котрі змінюються постійно. Років п'ять назад, коли нагромадження даних було проблемою для будь-якої інформаційної системи, дуже великою вважали базу даних, розмір якої перевищував 10 Гбайт, зараз цей критерій збільшений до 100 Гбайт.

Моделі побудови БД

Існує декілька методів побудови бази даних, але всі вони так чи інакше задовільняють вимогам до DBMS. Основні моделі організації баз даних наступні:

- **Лінійна** (**Flat-file**) чи плоска. Назвати базою даних її можна умовно, тому що скоріше це список або таблиця. Усі дані зберігаються у вигляді послідовного списку одноманітних записів в одному файлі. Такі бази досить громіздкі, а через послідовний спосіб доступу до записів ще й повільні.
- **Реляційна** (**Relational**). Побудована на взаємозв'язку декількох лінійних баз – таблиць. Починаючи з 70-х років це була найбільш популярна модель баз даних. До реляційних СУБД відносять Oracle, Sybase, Informix, MS Access і багато інших.
- **Ієрархічна** (**Hierarchical**). Дозволяє зберігати дані в ієрархічній структурі. Прикладом такої бази є **IMS** (**Information Management Systems – система керування інформацією**). У спадщину від колишніх часів нам дісталося досить багато баз даних, встановлених на міні-комп'ютерах і мейнфреймах, що використовували мережеву й ієрархічну моделі бази даних.

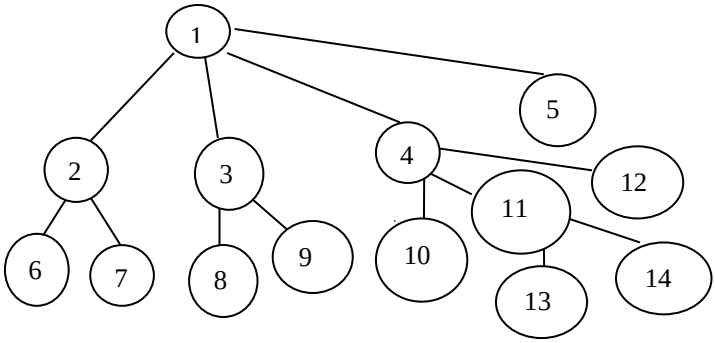
- **Мережева** (**Network**). Ця модель зберігає дані в структурі мережі передачі даних. Прикладом такої бази є **IDMS** (**Integrated Data Management Systems – інтегрована система керування базами даних**).
- **Об'єктно-орієнтована модель** (**Object-oriented**). Стала популярною останнім часом завдяки розвитку мережі Інтернет. На відміну від традиційних, побудованих по реляційній моделі, об'єктно-орієнтировані бази позбавлені обмежень щодо обробки розподілених даних з використанням транзакцій. У наш час такі бази застосовують також у пакетах прикладних програм, які широко використовують засоби мультимедіа чи оперують великою кількістю графічних даних, наприклад у пакетах автоматичного проектування CAD/CAM.

Реляційна модель БД. Реляційна модель БД представляє об`єкти та зв'язки між ними у вигляді таблиць, а всі операції над ними зводяться до операцій над цими таблицями. На цій моделі базуються практично всі сучасні СУБД. Ця модель більш зрозуміла, „прозора" для кінцевого користувача організації даних. До переваг реляційної моделі можна віднести також більш високу гнучкість при розширенні БД, складу запитів до неї. Реляційна організація БД у вигляді таблиці може містити, наприклад, програму випуску виробів. Ця база даних включає в себе три атрибути: код технологічної групи обладнання, код виробу, програму випуску.

Реляційна модель БД

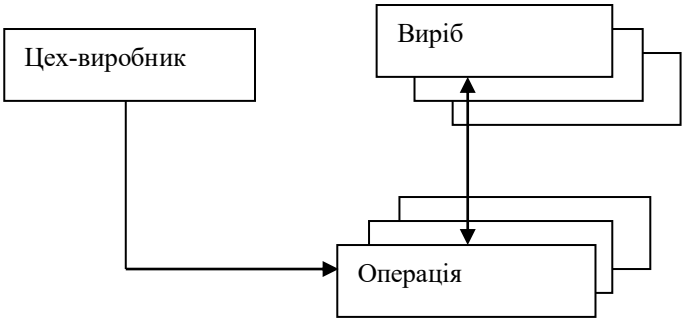
код технологічної групи обладнання	код виробу	програма випуску
3	20370	600
3	20510	2000
5	50200	1500
5	50230	300

Ієрархічна модель БД. Ієрархічну модель БД зображають у вигляді дерева.



Елементи дерева, вершини 1-14, є сукупністю даних, наприклад, логічних записів. Кожній вершині відповідає багато екземплярів записів, які складають логічний файл. Вершини розміщені по рівнях і пов'язані між собою відношенням підпорядкованості. Одна-єдина вершина верхнього рівня є кореневою. Ієрархічна модель даних забезпечує так зване одно-багатозначне відношення між даними. Прикладом таких відношень можуть служити наступні: одному виробу відповідає декілька матеріалів, які використовуються на різних операціях обробки, складання.

Мережева модель БД. Мережеві моделі БД відповідають більш ширшому класу об`єктів управління, хоча й потребують для своєї організації додаткових витрат. Мережева модель дозволяє будь-якому об'єкту бути пов`язаним з будь-яким іншим об'єктом. Мережеві моделі складні, що складає деякі труднощі при необхідності модернізації або розвитку СУБД. Приклад мережевої моделі БД представлений на рисунку, з якого видно, що один виріб виготовляється в результаті виконання декількох операцій, а одна операція може використовуватися для виготовлення різних виробів.



Об'єктно-орієнтовані бази даних. Останніми роками все більшого застосування і розвитку отримують об'єктно-орієнтовані бази даних (ООБД), поштовх до появи яких дали об'єктно-орієнтовані програми та використання ПК для обробки та представлення практично всіх форм інформації, яка сприймається людиною. В ООБД модель даних більш близька до суті реального світу, тому що об'єкти можна зберігати та використовувати безпосередньо, не розміщуючи їх по таблицях. Типи даних розрізняються розробником і не обмежуються набором визначених типів. В об'єктних СУБД дані про об'єкт, а також його методи поміщаються в сховище, як одне ціле. Об'єктна СУБД саме той засіб, який забезпечує запис об'єктів в базу даних. Суттєвою особливістю ООБД

можна назвати об`єднання об'єктно-орієнтованого програмування (ООП) з технологією баз даних для створення інтегрованого середовища розробки прикладних програм. ООБД забезпечують доступ до різних джерел даних, у тому числі, звичайно, і до даних реляційних СУБД, а також різні засоби маніпулювання з об'єктами баз даних. Традиційними областями застосування об`єктних СУБД є системи автоматизованого проектування (САПР), моделювання, мультимедіа, оскільки саме з потреб цих застосувань виріс новий напрям у розробці баз даних. В даних областях завжди існувала потреба знайти адекватний засіб збереження великих об`ємів різного роду даних. Оскільки об`єктні СУБД відрізняються високою швидкодією, надійністю, представляють різноманітний програмний інтерфейс для розробників, вони широко застосовуються у телекомунікаціях, різних засобах автоматизації підприємства, у видавничій справі, в геоінформаційних проектах. Об'єктні СУБД дуже добре підходять для вирішення задач побудови розподілених обчислювальних систем. На основі об'єктної СУБД можна будувати складні розподілені бази даних, організовувати до них доступ, як через локальну мережу, так і для віддалених користувачів в режимі реального масштабу часу. До об'єктних СУБД можна віднести СУБД Jasmine, Jupiter, ORACLE.

CASE-технологія в контексті баз даних

За останнє десятиліття в області засобів програмування сформувався новий напрям – **CASE-технологія** (**Computer-Aided Software/System Engineering**). CASE-технологія є сукупністю методологій аналізу, проектування, розробки та супроводу складних систем і підтримується комплексом взаємозв'язаних засобів інформаційних технологій. При використанні методологій структурного аналізу з'явився ряд обмежень (складність розуміння, велика трудомісткість і вартість використання, незручність внесення змін у проектні специфікації). CASE-технології з самого початку розвивалися саме з метою подолання цих обмежень шляхом автоматизації процесів аналізу та інтеграції підтримуючих засобів.

Сучасні CASE-технології охоплюють широку область підтримки численних технологій проектування БД: від простих засобів аналізу та документування, до повномасштабних засобів автоматизації, що покривають весь життєвий цикл БД. Найтрудомісткішими етапами розробки БД є етапи аналізу та проектування, в процесі яких CASE-технології забезпечують якість ухвалюваних технічних рішень і підготовку проектної документації. При цьому велику роль відіграють методи візуального представлення інформації. Це забезпечує побудову структурних або інших діаграм в реальному масштабі часу, використання багатообразної колірної палітри, наскрізну перевірку синтаксичних правил. Графічні засоби моделювання наочної області дозволяють розробникам в

наочному вигляді вивчати існуючу інформаційну систему, перебудовувати її відповідно до поставлених цілей і наявних обмежень.

Основні можливості CASE-засобів

У якнайповнішому виді CASE-технології мають наступні характерні особливості:

– єдину графічну мову. CASE-технології забезпечують усіх учасників проекту, включаючи замовників, єдиною строгою, наочною та інтуїтивно зрозумілою графічною мовою, що дозволяє одержувати компоненти БД з простою та ясною структурою. При цьому БД представляються двовимірними схемами (які простіші у використанні, ніж багатосторінкові описи), що дозволяють замовнику брати участь в процесі розробки, а розробникам – спілкуватися з експертами наочної області, розділяти діяльність системних аналітиків, проектувальників і програмістів, полегшуючи їм захист проекту перед керівництвом, а також забезпечуючи легкість супроводу та внесення змін в БД;

– єдина база даних проекту. Основа CASE-технології – використання бази даних проекту (репозиторія) для зберігання усієї інформації про проект, яка може спільно використовуватися розробниками відповідно до їх прав доступу. Вміст репозиторію включає не тільки інформаційні об'єкти різних типів, але й відносини між їх компонентами, а також правила використання або обробки цих компонентів. Репозиторій може зберігати об'єкти різних типів: структурні діаграми, визначення екранів і меню, проекти звітів, описи даних, логіку обробки, моделі даних, їх організації і обробки, початкові коди, елементи даних і т. ін.;

– інтеграція засобів. На основі репозиторія здійснюються інтеграція CASE-технології і розділення системної інформації між розробниками. При цьому можливості репозиторія забезпечують декілька рівнів інтеграції: загальний призначений для користувача інтерфейс по всіх засобах, передачу даних між засобами, інтеграцію етапів розробки через єдину систему представлення фаз життєвого циклу, передачу даних і засобів між різними платформами;

– підтримка колективної розробки та управління проектом. CASE-технології підтримують групову роботу над проектом, забезпечуючи можливість роботи у мережі, експорт/імпорт будь-яких фрагментів проекту для їх розвитку і/або модифікації, а також планування, контроль, керівництво та взаємодію, тобто функції, необхідні в процесі розробки та супроводу проектів. Ці функції також реалізовані на основі репозиторія. Зокрема, через репозиторій можна здійснювати контроль безпеки (обмеження і привілеї доступу), контроль версій і змін БД;

– макетування. CASE-технологія дає можливість швидко будувати макети (прототипи) майбутньої БД, що дозволяє замовнику на ранніх

етапах розробки оцінити, наскільки вона влаштовує його та прийнятна для майбутніх користувачів;

– генерація документації. Вся документація за проектом генерується автоматично на базі репозиторія (як правило, відповідно до вимог діючих стандартів). Безперечна перевага CASE-технології полягає у тому, що документація завжди відповідає поточному стану справ, оскільки будь-які зміни в проекті автоматично відбиваються в репозиторії (відомо, що при традиційних підходах до розробки програмного забезпечення документація в кращому разі запізнюється, а ряд модифікацій взагалі не знаходить в ній віддзеркалення);

– верифікація проекту. CASE-технологія забезпечує автоматичну верифікацію і контроль проекту щодо повноти та спроможності на ранніх етапах розробки, що впливає на успіх розробки в цілому;

– автоматична генерація програмного коду. Генерація програмного коду здійснюється на основі репозиторія і дозволяє автоматично побудувати до 85-90% текстів мовами високого рівня.

– супровід і реінжиніринг. Супровід системи в рамках CASE-технології характеризується супроводом проекту, а не програмних кодів. Засоби реінжинірінга та зворотнього інжинірингу дозволяють створювати модель системи з її кодів та інтегрувати одержані моделі в проект, автоматично оновлювати документацію при зміні кодів, автоматично змінювати специфікації при редагуванні кодів і т.ін.

Далеко не всі CASE-засоби підтримують усі вказані вище можливості. Тому звичайно до CASE-засобів відносять будь-який програмний продукт, який автоматизує ту або іншу сукупність процесів життєвого циклу БД і який має наступні основнні характерні особливості:

– наявність могутніх графічних засобів для опису та документування БД, що забезпечує зручний інтерфейс для розробників БД і розвиває їх творчі можливості;

– інтеграція окремих компонентів CASE-засобів, що забезпечує керованість процесу розробки БД;

– використання спеціальним чином організованого сховища проектних метаданих (репозиторія).

Концептуальне моделювання структури даних

Створення бази даних і вхідних в неї таблиць за допомогою SQL-конструкцій можна застосувати при конструюванні невеликих БД, але створення великих баз даних, що містять сотні чи тисячі таблиць і складні зв'язки між ними, можливо тільки при використанні CASE-технології. Вручну дуже важко розробити і представити графічно структуру системи, перевірити її на повноту та несуперечність, відстежувати версії і

виконувати модифікації. Процес проектування БД за допомогою CASE-технології починається з виділення деяких об'єктів (суті) наочної області, істотних для застосування, і виявлення зв'язків в рамках цієї суті. Для забезпечення потреби проектувальників баз даних в зручніших і могутніших засобах моделювання наочної області проектування баз даних звичайно виконується не в термінах реляційної моделі, а з використанням **концептуальних моделей** наочної області. Звичайно розрізняють концептуальні моделі двох видів:

– ***семантичні моделі БД***, які відображають значення реальної суті та відносин;

– ***об'єктно-орієнтовані моделі БД***, в яких суть реального світу представляється у вигляді об'єктів, а не записів реляційних таблиць.

Об'єктно-орієнтовану модель БД можна розглядати як результат об'єднання семантичної моделі даних і об'єктно-орієнтованої мови програмування. Не дивлячись на те, що останнім часом все більшого поширення набувають об'єктно-орієнтовані моделі, не знижується і значення семантичних моделей. Концептуальне моделювання баз даних на основі семантичних моделей підтримується у всіх відомих CASE-засобах. Крім того, семантичні моделі простіші для розуміння, особливо при проектуванні порівняно невеликих баз даних.

Як і реляційна модель, будь-яка розвинута семантична модель даних включає структурну, маніпуляційну та цілісну частини. Головним призначенням семантичних моделей є забезпечення можливості виразу семантики даних.

Мета **семантичного моделювання** – забезпечення найприродніших для розробника способів збору та представлення тієї інформації, яку передбачається зберігати у створюваній базі даних. Тому семантичну модель даних намагаються будувати за аналогією з природньою людською мовою (остання не може бути використана в чистому вигляді через складність комп'ютерної обробки текстів і неоднозначність будь-якої природньої мови). Основними конструктивними елементами семантичних моделей є суть, зв'язки між ними та їх властивості (атрибути).

Природа моделі „суть-зв'язок"

Однією з найпопулярніших семантичних моделей даних є **модель „суть-связок"** (часто її називають також ER-моделлю – за першими буквами англійських слів Entity (суть) і Relation (зв'язок)). На використанні різновидів ER-моделі засновано більшість сучасних підходів до проектування баз даних (головним чином, реляційних). Модель була запропонована професором Ченом у 1976 р. Моделювання наочної області базується на використанні графічних діаграм, які включають невелику кількість різнорідних компонентів. Через наочність представлення

концептуальних схем баз даних ER-моделі набули широкого поширення у CASE-засобах, призначених для автоматизованого проектування реляційних баз даних.

Для моделювання структури даних використовуються ER-діаграми (діаграми „суть-зв'язок"), які в наочній формі представляють зв'язки між сутями. Відповідно до цього ER-діаграми набули поширення в CASE-системах, які підтримують автоматизоване проектування реляційних баз даних. Основними поняттями ER-діаграми є **суть**, **зв'язок** та **атрибут**.

Суть – це реальний або віртуальний об'єкт, котрий має істотне значення для даної наочної області, інформація про який підлягає зберіганню. Можна вважати, що суть відповідає таблицям реляційної моделі. Кожна суть повинна володіти наступними властивостями:

1. Мати унікальний ідентифікатор.
2. Містити один або декілька атрибутів, які або належить суті, або успадковуються через зв'язок з іншою суттю.
3. Містити сукупність атрибутів, які однозначно ідентифікують кожен екземпляр суті.
4. Будь-яка суть може мати довільну кількість зв'язків з іншою суттю.

Зв'язок – це з'єднання двох сутей, при якому, як правило, кожен екземпляр однієї суті, званою батьківською суттю, асоційований з довільною (зокрема нульовою) кількістю екземплярів другої суті, званою суттю-нащадком, а кожен екземпляр суті-нащадка асоційований в точності з одним екземпляром суті-батька.

Атрибут – є характеристикою суті, значущої для даної наочної області. У ER-діаграмах список атрибутів суті відображається у вигляді рядків усередині прямокутника із зображенням суті. У реляційних базах даних аналогом атрибуту є поле таблиці.

Технологія створення фізичної моделі БД

Концептуальна модель БД дозволяє зрозуміти суть створюваної БД, але вона не підходить для створення безпосередньо структури бази даних. Для генерації структури бази даних необхідно перетворити концептуальну модель бази даних у фізичну. Загальні принципи такого перетворення полягають у наступному:

– кожна суть перетворюється у таблицю. Ім'я суті стає ім'ям таблиці;

– кожен атрибут стає стовбцем таблиці з тим же ім'ям, уточнюється тип даних, вибирається точніший формат;

– ідентифікуючі атрибути суті перетворюються на первинний ключ таблиці. Якщо для даної суті є залежні зв'язки, до стовбців первинного ключа додається копія унікального ідентифікатора суті, яка

знаходиться на іншому кінці зв'язку (цей процес може продовжуватися рекурсивно);

– зв'язки типу „багато-до-одного" та „один-до-одного" стають зовнішніми ключами. Для них створюється копія унікального ідентифікатора з одиночного зв'язку, та відповідні стовбці складають зовнішній ключ;

– для первинного ключа (унікальний індекс) і зовнішніх ключів створюються індекси;

– для зв'язків „багато-до-багатьох" створюється таблиця, стовбцями якої є унікальні ідентифікатори зв'язуваної суті (вони складають первинний ключ).

Після створення фізичної моделі та її уточнення можна створювати структуру бази даних за допомогою відповідних команд вікна діалогу CASE-засобів. Життєвий цикл створення і супроводу інформаційної системи має вигляд спіралі. Це означає, що модифікація структури бази даних практично неминуча. Використання CASE-засобів дозволяє значно полегшити підтримку декількох версій структури та автоматизувати створення сценаріїв зміни структури бази даних.

Концепція побудови сучасних баз даних

Бази даних створюють в організаціях і на підприємствах для вирішення за дпомогою комп'ютера задач управління виробництвом чи бізнесом. Будь-яка БД завжди має у своєму складі два основні компоненти: власне таблиці, у яких зберігають дані – базу даних, які є не що інше, як даталогічне уявлення інформаційної моделі організації чи підприємства, і систему управління базою даних (СУБД), за допомогою якої реалізують централізоване управління даними, що зберігаються в базі, доступ до них і підтримку їх в стані, що відповідає стану предметної області. Для ефективної роботи з БД створюють також допоміжні програми, наприклад, довідники про структуру БД, про розподіл файлів, в яких зберігають таблиці, для друку БД, для доступу до БД через мережу Інтернет. Така концепція бази даних забезпечує не тільки інтегроване зберігання даних, але й розмежування опису даних від програм їх обробки, інтерфейс з якими забезпечується СУБД. В основу розробок даної концепції покладені наступні принципи: єдиність структурно-інформаційної організації масивів, централізація процесів накопичення, зберігання та обробки різних видів інформації, одноразове введення первинних масивів інформації з наступними багаторазовим і багатоцільовим його використанням, інтегроване використання масивів в різних режимах обробки, оперативність доступу до різних елементів інформаційних масивів, мінімізація вартості створення і функціонування бази даних.

Системи управління базами даних

Для багатоцільового використання та ефективного задоволення інформаційних потреб різних користувачів, при збільшенні об'ємів інформації використовують інтегрований підхід до створення БД. При цьому дані розглядають як інформаційні ресурси для різноаспектного та багаторазового використання. Принцип інтеграції передбачає організацію збереження інформації за допомогою СУБД, а відтак, усі дані зібрані в єдиному інтегрованому сховищі та до інформації, як до найважливішого ресурсу забезпечений зручний доступ для широкого кола різноманітних користувачів. Таким чином СУБД – це система програмних, технічних, мовних, організаційно-методичних засобів, призначених для забезпечення централізованого накопичення і колективного багатоцільового використання спеціальним чином організованих даних (баз даних).

Основною вимогою до сучасних СУБД є: інтегрованість баз даних і цілісність кожної з них; незалежність, мінімальна збитковість даних, що зберігаються і можливість їх розширення (модифікації). Важливою умовою ефективного функціонування СУБД є забезпечення захисту даних від несанкціонованого доступу або випадкового знищення даних, які в них зберігаються. СУБД – це пакет програм, які забезпечують пошук, зберігання, коректування даних, формування відповідей на запити. СУБД забезпечує збереження даних, їх конфіденційність, переміщення і зв'язок з іншими програмними засобами. Основні функції СУБД наступні: безпосереднє управління даними всередині пам`яті, управління буферами оперативної пам'яті, управління транзакціями, ведення журналу БД. Організація типової СУБД і склад її компонентів відповідає розглянутому набору функцій. Логічно у сучасній реляційній СУБД можна виділити найбільшу внутрішню частину – ядро СУБД, компілятор мови БД (зазвичайно, це SQL), підсистему підтримки БД під час виконання різних функцій, набір утиліт.

Переваги роботи з СУБД для користувачів окуповують затрати на її створення або придбання. Вони полягають в наступному: підвищується продуктивність роботи користувачів, досягається ефективне задоволення інформаційних потреб, централізоване управління даними звільняє прикладних програмістів від роботи щодо організації даних, забезпечує незалежність прикладних програм від даних. СУБД дозволяє реалізувати інші нерегламентовані запити, знижуються затрати не тільки на створення і зберігання даних, але й на підтримання їх в актуальному динамічному стані, зменшуються потоки даних, що циркулюють в інформаційній системі, зменшується збитковість і дублювання БД.

Основні технології організації та обробки даних

Щодо організації та обробки даних технології баз даних поділяються на два види: централізовані та розподілені. Централізована база даних має традиційну архітектуру бази даних, таку що всі необхідні для роботи спеціалістів дані і СУБД розміщені на центральному комп'ютері або мейнфреймі (англ. mainframe), разом з програмою, яка приймає вхідну інформацію призначеного для користувача терміналу та відображає дані на екрані користувача. Так що у випадку коли користувач вводить запит, який потребує послідовного перегляду бази даних (наприклад, запит на розрахунок потреби в матеріалах на деталь у натуральному і вартісному вираженні), СУБД отримує цей запит, переглядає БД, вибирає з диска потрібний запис, обраховує значення і відображає результат на екрані. Програма та СУБД працюють на одному комп'ютері і, оскільки програма обслуговує багато користувачів, кожний з них відчуває зниження швидкодії в міру збільшення навантаження на систему.

Розподілена база даних складається з декількох, можливо перехресних або навіть таких, що повторюють одна одну, частин, які зберігаються в різних комп'ютерах, об'єднаних за допомогою мережі передачі даних. Робота з такою БД здійснюється за допомогою системи управління розподіленою базою даних (СУРБД).

Природа БД з локальним доступом і БД з мережевим доступом

За способом доступу до даних розподілені БД розділяються на БД з локальним доступом і БД з мережевим доступом. Системи централізованих БД з мережевим доступом передбачають різні архітектури інформаційних систем: „файл-сервер" і „клієнт-сервер". Поява персональних комп'ютерів і мережі передачі даних призвела до розробки архітектури „файл-сервер". При такій архітектурі програма, яка виконується на ПК, може одержати повний доступ до „файл-сервера", на якому зберігаються файли сумісного користування. Коли програмі, яка працює на ПК необхідно одержати дані із файла спільного користування, мережеве програмне забезпечення автоматично зчитує потрібний блок даних з сервера. Найпопулярніші СУБД для ПК, включаючи Microsoft Access, Paradox і dBASE, підтримують архітектуру „файл-сервер", при котрій на кожному ПК працює своя копія СУБД. При виконанні звичайних запитів ця архітектура забезпечує високу продуктивність, оскільки в наявності кожної копії СУБД знаходяться всі ресурси ПК. Однак, оскільки запит потребує послідовного перегляду БД, СУБД постійно запитує все нові блоки даних БД, яка фізично розміщена на сервері мережі. Очевидно, що в результаті СУБД запитає і одержить по мережі всі блоки файлів. При виконанні запитів такого типу ця архітектура створює дуже велике навантаження на мережу та зменшує продуктивність її роботи.

При використанні архітектури „файл-сервер” ПК об’єднані в локальну мережу, в котрій є сервер баз даних, в якому заходиться загальна база даних. Функції СУБД розділені на дві частини. Програми користувача, такі як програми для формування інтерактивних запитів і генератори звітів, працюють на комп’ютері користувача. Збереження даних і управління ними забезпечується сервером. У цій архітектурі мова SQL стала стандартною мовою, призначеною для обробки та зчитування даних, наявних у БД. SQL забезпечує взаємодію між програмами користувача та ядром БД. Розглянемо для прикладу визначення потреби матеріалів на деталь. При архітектурі „клієнт-сервер” запит передається по мережі на сервер БД у вигляді SQL-запитів. Ядро БД на сервері опрацьовує запит і переглядає БД, яка також розміщена на сервері. Після обчислення результатів ядро БД посилає його програмі клієнта, яка відображає його на екрані ПК. Архітектура „клієнт-сервер” дозволяє зменшити трафік і розподілити процес завантаження бази даних. Функції роботи з користувачем, такі, як обробка, введення і відображення даних, виконуються на ПК користувача. Функції роботи з даними, такі як дискове введення/виведення і виконання запитів, виконуються сервером БД. Найбільш важливо тут те, що SQL забезпечує чітко визначений інтерфейс між користувацькою і серверною системами, ефективно передаючи запити на доступ до БД. Ця архітектура використовується в сучасних СУБД ORACLE, Informix, Sybase.

Цілі послуговування СУБД

Досить часто бази даних об’єднують в одному програмному продукті разом із **системою управління базою даних (СУБД) (Database Management System – DBMS),** як, наприклад, Microsoft Access. Але в більшості випадків доцільно мати систему управління базою даних у вигляді окремого програмного продукту, призначеного для роботи з базами даних. До такої програми пред’являють ряд специфічних вимог, але тільки найсучасніші системи здатні забезпечити їх у повному обсязі:

- СУБД повинна мати простий доступ до даних у базі.
- СУБД повинна забезпечувати захист даних від несанкціонованого доступу.
- Повинні бути забезпечені можливості паралельного користувацького доступу та контролю процесу доступу.
- Повинен існувати метод відновлення даних бази у випадку пошкодження системи.
- Повинні існувати методи перевірки „парності” та підтримки цілісності даних.
- Метадані бази повинні також бути доступні для користувачів. Метадані є системною інформацією бази, яка описує збережені дані; інакше

кажучи, каталог, чи зміст вмісту бази: інформацію про назви полів, їхні типи, місця розташування і про всі інші дані, що містяться в базі.

Вимоги до побудови СУ реляційними БД?

На початку розвитку реляційних баз даних **доктор Е.Ф. Кодд** (**Dr. E.F. Codd**) вніс важливий вклад у принципи їх побудови. Він запропонував 12 критеріїв, яким повинна задовільняти СУБД (DBMS), щоб базу даних можна було вважати насправді реляційною (пізніше ці критерії були доповнені та деталізовані):

1. Інформація повинна бути представлена у вигляді даних, які можна зберігати в окремих комірках.
2. Для доступу до будь-якого елементу даних повинна застосовуватися наступна комбінація:
   - ім'я таблиці;
   - первинний ключ таблиці;
   - ім'я стовпця таблиці.
3. Комірка, яка не містить жодного значення, обов'язково повина бути заповненою значенням NULL. Якщо в полі для числа чи для символа значення NULL неприпустиме, то необхідно застосовувати значення за замовчуванням.
4. Вбудований словник даних повинен підтримуватися стандартною мовою доступу до даних і зберігатися як реляційна таблиця.
5. **Мова доступу до даних** (**Data Access Language**) повинна бути єдиним способом доступу до даних у СУБД.
6. Для даних, які змінюються, повинен існувати механізм відновлення значень.
7. Повинні підтримуватися операції вставки, створення і видалення (знищення) даних.
8. СУБД не повинні залежати від фізичних атрибутів даних. Якщо, наприклад, файл у якому записана певна таблиця переміщається з одного диска на інший, то це не повинне вимагати внесення змін у СУБД.
9. СУБД не повинні залежати від логічних атрибутів даних. Якщо таблиця, наприклад, розбивається на дві, то все одно повинен існувати спосіб доступу до даних, а також спосіб відновлення первісної таблиці.
10. Правила цілісності даних, такі як властивості первинних і зовнішніх ключів, повинні зберігатися у словниках даних.
11. Повинна існувати можливість розміщення бази даних як локально, так і на розподілених комп'ютерах.
12. Жоден низкорівневий спосіб доступу, наприклад резервне копіювання чи утиліта завантаження, не повинні обійти стандартну систему безпеки бази даних.

РОЗДІЛ 2. ТЕХНОЛОГІЯ РОБОТИ З БАЗАМИ ДАНИХ В СЕРЕДОВИЩІ ЕЛЕКТРОННИХ ТАБЛИЦЬ

Таблична база даних Excel

Багато користувачів Excel застосовують цю програму для роботи зі **списками**, або по-іншому, **базами даних робочого аркуша**.

Говорячи по суті, **таблична база даних** – це впорядкований набір даних. Зазвичай база даних робочого аркуша складається із **рядка заголовків** (опису даних) і **рядків даних**, які можуть бути числовими або текстовими.

Стовбці бази даних робочого аркуша часто називають **полями**, а рядки – **записами**. Розмір табличної бази даних обмежений розмірами одного робочого аркуша. По-іншому, таблична база даних може мати не більше 256 полів і не більше 65535 записів (один рядок містить заголовки полів). Очевидно, що створення таких баз даних і подальша робота з ними є непродуктивною. З іншого боку, таблична база даних може складатися із двох комірок (назва поля, під яким знаходиться одна комірка з даними). Хоча користі від табличної бази даних з двох комірок небагато, вона попри те розглядається як база даних.

Цілі послуговування базою даних робочого аркуша

В Excel є декілька засобів, призначених для роботи із табличними базами даних. Найуживанішими операціями для роботи із табличними базами даних є:

- Введення даних у базу.
- Фільтрування бази даних для вибіркового відображення рядків (за певним критерієм).
- Сортування бази даних.
- Вставка формул для підбиття проміжних підсумків.
- Створення формул для обчислення результатів у базі даних, яка відфільтрована за певними критеріями.

Правила побудови табличних баз даних

Наведемо основні рекомендації, яких варто дотримуватися створюючи табличні бази даних.

- Розміщувати заголовки (по одному для кожного стовбця) в першому рядку табличної бази даних, який має назву рядок заголовків. Якщо заголовки є довгими, використовувати текстовий формат з перенесенням слів, – тоді не доведеться розширювати стовбці.
- Слідкувати, щоб у кожному стовбці містилася однотипна інформація. Наприклад, не слід змішувати в одному стовбці дати та звичайний текст.

- Можна застосовувати формули, які використовують значення з інших полів цього ж запису. Якщо формула посилається на комірку, розташовану поза списком, потрібно зробити посилання на цю комірку абсолютним, по-іншому результати сортування списку можуть бути непередбачуваними.
- Не використовувати порожніх рядків у табличній базі даних. При проведенні операцій над табличною базою даних Excel визначає її межі автоматично, при цьому порожній рядок означає кінець табличної бази даних.
- Послуговуватися командою **Закріпити області/Freeze Panes** на вкладці **Подання/View**, щоб заголовки було завжди видно під час прокручування аркуша табличної бази даних.
- Завжди намагатися попередньо форматувати весь стовбець, щоб дані мали один і той самий формат. Наприклад, якщо стовбець місить дати, вибирають необхідний формат для відображення дат у цьому стовбці.

Способи введення даних у табличну базу даних

Дані можна ввести у табличну базу даних трьома способами:

- Вручну, послуговуючись стандартними методами введення даних.
- Імпортувати чи скопіювати з іншого файлу.
- Використовувати форму введення.

В операції введення даних у табличну базу даних немає нічого особливого. Переміщуючись по таблиці, водять дані у потрібні комірки.

В Excel є два засоби, які допомагають уникнути виконання нудної операції введення одноманітних даних.

- **Автозаповнення.** Коли починають вводити дані, програма переглядає стовбець, щоб вияснити, чи зможе вона впізнати те, що набирають. Якщо Excel знаходить закономірність, то заповнює залишок комірок автоматично. Щоб завершити введення даних, натискують клавішу <Enter>. Можна вмикати чи вимикати цю можливість за допомогою опції **Автозавершення значень клітинок/Enable AutoComplete for sell values** у групі **Параметри редагування/Editing options** розділу **Додатково/Advanced** у діалоговому вікні **Параметри Excel/Excel Options**, яке з'являється, якщо на вкладці **Файл/File** вибрати пункт **Параметри/Options**.
- **Вибір зі списку.** Цокнути правою кнопкою мишки на комірці та вибрати із контекстного меню, яке з'явилося, команду **Вибрати з розкривного списку…/Pick From List…** . З'явиться список з усіма елементами, котрі знаходяться у стовбці. Вибирають зі списка потрібний елемент, і він буде відображений у комірці (при цьому нічого набирати не потрібно!).

Послуговування формою введення даних

Щоб вивести на екран форму для введення даних, табличний курсор розміщують у якому-небудь місці табличної бази даних та вибирають команду **Форма…/Form…** на панелі швидкого доступу. Ця ж команда може знаходитися і на стрічці[2]. Програма встановить розмір табличної бази даних та виведе діалогове вікно, у якому буде знаходитись кожне поле бази даних. Поля, котрі містять формулу, у формі введення відображаються, але їх значення неможливо змінити.

Коли з'явиться форма, у ній буде показаний перший запис табличної бази даних. Індикатор у правому верхньому кутку форми показує номер вибраного запису та загальну кількість записів у табличній базі даних.

Щоб ввести новий запис, потрібно цокнути на кнопці **Створити/New**, а відтак, у формі очистяться всі поля. Тепер можна вводити нову інформацію у відповідні поля. Для переміщення від одного поля до іншого послуговуються клавішами <Tab> або <Shift+Tab>. Після цокання на кнопці **Створити/New** або **Закрити/Close** введені дані з'являться у кінці бази даних. Можна також натиснути клавішу <Enter>, що еквівалентно цоканню на кнопці **Створити/New**. Якщо база даних містить формули, то вони автоматично з'являться у нових записах.

Інші застосування форми введення

Форму введення можна використовувати не тільки для введення даних. Вона дозволяє редагувати існуючі записи, переглядати їх, видаляти та вибірково відображати записи за деяким критерієм.

Форма введення містить декілька кнопок.

- **Видалити/Delete.** Видаляє поточний запис.
- **Відновити/Restore.** Відміняє усі внесені у поточний запис зміни. Ця кнопка працює доти, допоки користувач не цокне на кнопці **Створити/New**.
- **Знайти наза/Find Prev.** Здійснюється перехід до попереднього запису табличної бази даних. Якщо встановлений критерій відбору, то відбудеться перехід до попереднього запису, який задовільняє даному критерію.
- **Знайти далі/Find Next.** Здійснюється перехід до наступнього запису табличної бази даних. Якщо встановлений критерій відбору, то відбудеться перехід до наступнього запису, який задовільняє даному критерію.
- **Умови/Criteria.** Очищує поля для введення критерію, за яким будуть відбиратися записи.

[2] Для того, щоб мати можливість виконати команду **Форма/Form**, необхідно спочатку додати цю команду на панель швидкого доступу або на стрічку.

- **Закрити/Close.** Закриває форму та записує введені дані в робочий аркуш.

Способи фільтрування табличних баз даних в Excel

Фільтрування табличної бази даних – це процес заховування усіх рядків, окрім тих, які задовільняють певним критеріям. Фільтрування – досить розповсюджена та дуже корисна операція. В Excel табличні бази даних можна фільтрувати двома способами:

- **Автофільтр** використовують для фільтрування за простими критеріями.
- **Розширений фільтр** застосовують для фільтрування за складнішими критеріями.

Процедура автоматичного фільтрування

Щоб автоматично відфільтрувати табличну базу даних, спочатку встановлюють табличний курсор на одну із комірок бази даних. Далі виконують команду **Фільтр/Filter**, яка розташована на вкладці **Дані/Data**. Excel проаналізує табличну базу даних і додасть у рядок заголовків полів кнопки списків, які розкриваються (кнопки автофільтру). Після цокання на одній із цих кнопок відкриється список, який міститиме усі унікальні елементи вибраного стовбця. Далі вибирають потрібний елемент. В результаті Excel заховає усі рядки, які не містять вибраного елемента. По-іншому, таблична база даних буде відфільтрована за вибраним елементом.

Після фільтрування табличної бази даних, у рядку стану з'явиться повідомлення про те, скільки рядків відібрано. Окрім того, зміниться колір кнопки списку, який розкривається, щоб нагадати користувачеві, що база даних відфільтрована за значеннями, котрі містяться у цьому стовбці.

Автоматичне фільтрування має обмеження. У списку, який розкривається, з'являються тільки 1000 перших різних значень. Якщо кількість елементів в списку перевищує вказану межу, то можна використовувати засоби розширеного фільтрування.

Окрім усіх значень стовбця, список, який розкривається, містить команду **Фільтри чисел/Number Filters**[3]. Ось тлумачення окремих її підкоманд.

- **Перші 10…/Top 10… .** Вибирає „кращу десятку" елементів табличної бази даних.
- **Більше середнього/Above Average.** Повертає ті значення поля, які більші середнього арифметичного.
- **Менше середнього/Below Average.** Повертає ті значення поля, які менші середнього арифметичного.

[3] Ця команда є контекстно-залежною, а саме, у випадку числових значень поля її назва дійсно **Фільтри чисел/Number Filters**, а наприклад, у випадку текстових значень, її назва **Текстові фільтри/Text Filters**.

- **Користувацький фільтр…/Custom Filter… .** Дозволяє фільтрувати табличну базу даних за декількома умовами.

Щоб уся таблична база даних була відображена знову, потрібно цокнути на кнопці розкриття списку та вибрати опцію **(Виділити все)/(Select All)**, або виконати команду **Очистити/Clear**, яка розташована в групі **Сортування й фільтр/Sort & Filter** вкладки **Дані/Data**.

Для скасування режиму Автофільтр і видалення кнопок списків, які розкриваються в іменах полів, вибирають команду **Фільтр/Filter** ще раз. В результаті таблична база даних робочого аркуша повернеться у звичайний стан.

Відфільтровування табличної бази даних за значеннями, які знаходяться у двох чи більше стовбцях

Інколи виникає необхідність фільтрування табличної бази даних за значеннями, котрі знаходяться у двох чи більше стовбцях. Для того щоб виконати це завдання, потрібно спочатку відфільтрувати базу даних за значенням з одного стовбця, а далі виконати фільтрування утвореного списку за значенням з іншого стовбця. В результаті буде відфільтрована вже раніше відфільтрована база даних. Отже, таблична база даних буде відфільтрована за значеннями у двох стовбцях. Можна фільтрувати табличну базу даних за будь-якою кількістю стовбців.

Природа користувацького автофільтра

Зазвичай автоматичне фільтрування полягає у виборі одного значення в одному чи декількох стовбцях. Якщо вибрати підкоманду **Користувацький фільтр…/Custom Filter…** команди **Фільтри чисел/Number Filters** у списку, який розкривається, то можна здійснити фільтрування гнучкішим способом. При виборі зазначеної опції з'явиться діалогове вікно **Користувацький автофільтр/Custom AutoFilter**. Це вікно дозволяє, зокрема, фільтрувати табличні бази даних з використанням декількох критеріїв:

- **Значення більше чи менше встановленого.** Наприклад, можна вибрати записи, котрі вказують на річні оклади, які перевищують 50000 умовних одиниць.
- **Значення в інтервалі.** Наприклад, потрібно відібрати всі записи, котрі вказують на об'єми продаж, які перевищують 10000 і не перевищують 25000 умовних одиниць.
- **Два окремих значення.** Наприклад, можна відібрати записи, в яких знаходиться інформація про співробітників, котрі працюють у Києві чи у Москві.
- **Вибір за шаблоном.** Можна використовувати символи підстановки „*" та „?", щоб відфільтрувати список гнучкішим способом. Символ „*" може замінити будь-яку кількість символів, а символ „?" – один символ. Наприклад, щоб вивести на екран

записи тільки про тих співробітників, прізвища яких починаються з букви В, використовують шаблон В*.

Користувацький автофільтр може бути достатньо корисним засобом, але він має певні обмеження. Наприклад, за допомогою нього неможливо відфільтрувати табличну базу даних за трьома чи більше значеннями (зокрема, вивести записи, в яких подана інформація про службовців, які працюють у містах Київ, Москва та Варшава). Для виконання задач такого типу необхідно послуговуватися засобом розширеного фільтрування.

Відфільтровування табличної бази даних так, щоб показати найбільші чи найменші значення

Інколи потрібно відфільтрувати числові поля так, щоб показати на екрані найбільші чи найменші значення. Наприклад, потрібно підготувати список, який складається із 12 співробітників, які працюють на фірмі найдовше. Вихід із цієї ситуації простий – скористатися командою **Фільтри чисел→Перші 10…/Number Filters→Top 10…** .

Назва Top 10 – це просто загальноприйнятий термін, дія ж опції не обмежується пошуком тільки 10-ти „найбільших" елементів. Насправді її дія навіть не обмежується пошуком тільки найбільших елементів. При виборі підкоманди **Перші 10…/Top 10…** з'являється діалогове вікно **Автофільтр для добору найкращої десятки/Top 10 AutoFilter**. За допомогою нього можна вибрати найбільші чи найменші елементи з табличної бази даних, а також вказати їх кількість. Наприклад, якщо необхідно встановити 12 співробітників, які пропрацювали в компанії найдовше, потрібно вибрати опцію **найменших/Bottom** і задати число 12. Після фільтрування база даних міститиме 12 записів, у полі **Дата призначення** яких містяться найменші значення. У цьому ж діалоговому вікні також можна вибрати опцію елементів **списку/Items** або **% від кількості елементів/Percent** від кількості елементів. Наприклад, можна відібрати 5% найбільших елементів табличної бази даних.

Копіювання та видалення відфільтрованих даних

Деякі стандартні операції робочого аркуша для табличних баз даних працюють по-іншому. Наприклад, нехай деякі рядки табличного документа були заховані за допомогою команди **Формат→Приховати або відобразити→Приховати рядки/Format→Hide & Unhide→Hide Rows** вкладки **Основне/Home**. Якщо далі скопіювати діапазон, котрий містить заховані рядки, то будуть скопійовані всі дані, включаючи й ті, які знаходяться у захованих рядках. Але якщо скопіювати дані із відфільтрованої бази даних, то будуть скопійовані тільки видимі комірки.

Аналогічним чином можна видаляти видимі рядки табличної бази даних, не зачіпаючи рядків, захованих за допомогою засобу Автофільтр.

Що дозволяє виконати розширене фільтрування

Часто для обробки табличної бази даних буває цілком достатньо автофільтра. Але якщо необхідно виконати операцію, яка виходить за рамки можливостей автофільтра, то доводиться звертатися за допомогою до засобів розширеного фільтрування. Розширене фільтрування значно гнучкіше ніж автофільтр, проте при його використанні потрібно виконати більше підготовчих дій. Розширене фільтрування дозволяє виконати таке:

- Визначити складніший критерій фільтрування.
- Встановити обчислювальний критерій фільтрування.
- Відображати тільки різні (які не повторюються) записи.
- Переміщувати копії рядків, які відповідають певному критерію, в інше місце.

Установлення діапазону критеріїв

Перед тим як послуговуватися засобом розширеного фільтрування, необхідно задати діапазон критеріїв. Це спеціально відведена область робочого аркуша, яка відповідає певним вимогам:

- Складається щонайменше із двох рядків, перший із яких повинен містити усі чи деякі назви полів табличної бази даних. Винятком є випадок, коли застосовується обчислювальний критерій, який може використовувати порожній рядок заголовків.
- Решта рядки повинні містити критерії фільтрування.

Незважаючи на те, що можна відвести для діапазону критеріїв будь-яке місце на робочому аркуші, бажано не задійувати при цьому рядків, котрі використовуються табличною базою даних. Оскільки деякі із рядків бази даних будуть заховані в результаті фільтрування, може виявитися, що діапазон критеріїв став невидимим після фільтрування. Тому діапазон критеріїв доцільно розміщувати над чи під списком.

Поля у кожному рядку діапазону (включаючи і рядок заголовків) з'єднані оператором **І/AND**.

Щоб виконати фільтрування, на вкладці **Дані/Data** вибирають команду **Додатково/Advanced**. З'являється діалогове вікно **Розширений фільтр/Advanced Filter**. Зазначають діапазон табличної бази даних, діапазон критеріїв і переконуються, що встановлений перемикач **фільтрувати список на місці/Filter the list, in-place**. Цокають на кнопці **ОК**, і база даних буде відфільтрована за заданими критеріями.

Природа множинного критерію відбору

Якщо в діапазоні критеріїв використовується декілька рядків, то критерії у кожному рядку з'єднані оператором **АБО/OR**. Діапазон

критеріїв може мати будь-яку кількість рядків, з'єднаних один з одним оператором **АБО/OR**.

Типи критеріїв

Записи у діапазоні критеріїв можуть бути двох видів:

- **Текстові або числові критерії.** Під час фільтрування для встановлення видимих записів здійснюється порівняння чисел чи рядків за допомогою операторів, таких як дорівнює (=), більше ніж (>), не дорівнює (<>) і т. ін.
- **Обчислювальні критерії.** Під час фільтрування виконуються деякі обчислення.

У критеріях, які містять текст, можна використовувати два символи підстановки: „*", яка замінює декілька символів, і „?", який використовують замість одного символа.

Природа обчислювальних критеріїв

Послуговування обчислювальними критеріями може зробити операцію фільтрування гнучкішою та потужнішою. Обчислювальний критерій ґрунтується на одному чи декількох обчисленнях і, по суті, обчислює для табличної бази даних нове поле. Через те потрібно розміщувати нову назву поля у першому рядку діапазону критеріїв, хоча використання заголовків полів для обчислювальних критеріїв необов'язкове.

При використанні обчислювальних критеріїв варто дотримуватися деяких правил:

- Не використовувати заголовки полів табличної бази даних у діапазоні критеріїв для обчислювального значення. Створити новий заголовок або просто залишити порожню комірку.
- Можна використовувати будь-яку кількість обчислювальних критеріїв, а також поєднання обчислювальних критеріїв з необчислювальними.
- Не звертати уваги на значення, які повертають формули у діапазоні критеріїв. Вони посилаються на перший рядок бази даних.
- Якщо обчислювальна формула посилається на значення поза списком, то потрібно використовувати абсолютні, а не відносні посилання.
- При створенні формул обчислювальних критеріїв використовувати перший рядок бази даних (не рядок заголовків!). Використовувати відносні, а не абсолютні посилання.

Інші можливості розширеного фільтрування

У діалоговому вікні **Розширений фільтр/Advanced Filter** є ще дві опції:

- **скопіювати результат до іншого розташування/Copy to another location.**
- **Лише унікальні записи/Unique records only.**

Якщо вибрати перемикач **скопіювати результат до іншого розташування/Copy to another location** у діалоговому вікні **Розширений фільтр/Advanced Filter**, то відібрані рядки будуть скопійовані в інше місце активного робочого аркуша або на інший аркуш. Місце визначається у полі **Діапазон для результату/Copy to**. Звертаємо увагу, що при використанні цієї опції сама таблична база даних не фільтрується.

При виборі опції **Лише унікальні записи/Unique records only** усі одинакові рядки, які відповідають встановленому критерію, будуть заховані. Якщо не встановити діапазон критеріїв, то у табличній базі даних будуть заховані всі одинакові рядки.

Послуговування функціями баз даних

Важливо завжди пам'ятати, що у формулах робочого аркуша використовуються усі вказані комірки, у тому числі й заховані в результаті операції фільтрування. Через те, якщо застосовувати наприклад, формулу **SUM**, за допомогою якої обчислюється сума значень у стовбці табличної бази даних, то вона поверне те ж саме значення незалежно від того, чи буде таблична база даних відфільтрована чи ж буде видно усі записи.

Щоб створити формулу, яка б повертала значення залежно від вибраних критеріїв фільтрування доцільно скористатися функціями із категорії баз даних Excel.

Наприклад, можна створити формулу для обчислення суми значень у базі даних, яка відповідає певному критерію. Для цього потрібно задати діапазон критеріїв і скористатися формулою такого типу:

=DSUM(ДіапазонБазиДаних;НазваПоля;Критерій)

У даному випадку замість аргумента **ДіапазонБазиДаних** підставляється посилання на базу даних, замість аргумента **НазваПоля** – посилання на комірку із заголовком поля того стовбця, значення якого будуть додаватися, а замість аргумента **Критерій** підставляється посилання на діапазон критеріїв.

Розглянемо найуживаніші функції Excel з категорії баз даних.

Функція	Опис
DAVERAGE	Повертає середнє значення вибраних елементів у базі даних
DCOUNT	Підраховує кількість комірок, які містять числа у певному полі бази даних, вибраних за заданими критеріями

DMAX	Повертає максимальне значення у вибраних записах бази даних
DMIN	Повертає мінімальне значення у вибраних записах бази даних
DPRODUCT	Перемножує значення у вказаному полі бази даних тих записів, які задовільняють встановленим критеріям
DSUM	Додає числа у вказаному полі всіх записів бази даних, які задовільняють заданим критеріям

Сортування табличної бази даних

Зазвичай порядок рядків у табличній базі даних не має значення. Проте інколи необхідно, щоб рядки мали певну послідовність. Так, наприклад, щоб якусь назву товару легше було знайти у базі даних, рядки повинні бути розташовані в абетковому порядку за назвами товарів.

Зміна порядку рядків у табличній базі даних називається сортуванням. Excel – дуже гнучка система щодо методів сортування даних, через те часто для виконання розглядуваної операції буває достатньо зробити усього одне цокання мишкою.

Виконання простого сортування

Щоб швидко відсортувати табличну базу даних у порядку зростання, переміщують табличний курсор на початок стовбця, по якому потрібно виконати сортування. Далі вибираємо команду **Сортування від А до Я/ Sort Ascending**, розташованій на вкладці **Дані/Data**. Команда **Сортування від Я до А/Sort Descending** працює так само, але база даних сортується у порядку спадання. В обидвох випадках Excel самостійно встановлює розмір табличної бази даних і сортує усі рядки у ній.

Сортування відфільтрованої бази даних виконується тільки для видимих рядків. Якщо відмінити фільтрування, то список виявиться невідсортованим.

Потрібно бути уважним під час сортування табличної бази даних з формулами. Якщо у формулах використовуються значення комірок, котрі знаходяться у тих самих рядках, то жодних проблем не виникне. Проте, якщо у формулах використовуються значення комірок, котрі знаходяться в інших рядках бази даних, то після сортування ці формули виявляться неправильними. Якщо формули у списку пов’язані з комірками поза табличною базою даних, потрібно щоб були вказані абсолютні адреси цих комірок.

Виконання складного сортування

В окремих випадках необхідно виконати сортування за двома чи більше стовбцями. Така операція проводиться, якщо сортування за одним полем залишає невідсортованими записи, які відповідають одинаковим значенням у полі, яке сортують.

Якщо необхідно виконати сортування за декількома полями, то на вкладці **Дані/Data** вибираємо команду **Сортувати/Sort** – з'являється діалогове вікно **Сортування/Sort**. У розкриваючому списку **Сортувати за/Sort by** вибираємо поле та вказуємо порядок сортування (**Від найменшого значення до найбільшого/Smallest to Largest** або **Від найбільшого значення до найменшого/Largest to Smallest**). Далі виконуємо теж саме і для другого поля. Якщо необхідно відсортувати і за третім полем, встановлюють третє поле у третьому розділі. Коли у розділі **My list has** встановлений перемикач **Header row**, то рядок заголовків полів не буде впливати на сортування. Цокаємо на кнопці **OK**, а відтак, рядки швидко перегрупуються.

Якщо результат сортування виявився незадовільним, на панелі швидкого доступу вибираємо команду **Скасувати Сортування/Undo Sort** або натискуємо комбінацію клавіш <Ctrl+Z>, щоб повернути базу даних у початковий стан.

Створення проміжних підсумків

Досить часто на практиці доводиться створювати проміжні підсумки. В Excel є засіб, який дозволяє виконати цю операцію автоматично. Щоб послуговуватися цією можливістю, таблична база даних повинна бути попередньо відсортована, оскільки проміжні підсумки будуть створюватися кожен раз при зміні значення певного поля.

Щоб формули проміжних підсумків вставлялись у базу даних автоматично, розміщують табличний курсор де-небудь у базі даних і на вкладці **Дані/Data** вибирають команду **Проміжні підсумки/Subtotal**. З'являється діалогове вікно **Проміжні підсумки/Subtotal**.

У цьому діалоговому вікні подано декілька опцій:

- **При кожній зміні в/At each change in.** У цьому розкриваючому списку показані всі поля табличної бази даних. Вибране поле повинне бути відсортованим.
- **Використовувати функцію/Use function.** Подані на вибір 11 функцій. Зазвичай використовують функцію **Сума/Sum**. Вона й задана за замовчуванням.
- **Додати підсумки до/Add subtotals to.** У цьому вікні виводяться назви всіх полів табличної бази даних. Цокаємо поруч з полем чи полями, в яких потрібно підвести проміжні підсумки.

- **Замінити поточні підсумки/Replace current subtotals.** Якщо встановити цю опцію, то будь-які існуючі формули підсумків будуть замінені новими.
- **Кінець сторінки між групами/Page break between groups.** Excel вставляє символ кінця сторінки після підведення кожного проміжного підсумку.
- **Підсумки під даними/Summary below data.** Якщо вибрана ця опція (вона встановлена за замовчуванням), то підсумки будуть розміщені під поточними даними. У протилежному випадку спочатку буде розміщений підсумок, а далі дані.

Послуговування кнопкою **Видалити все/Remove All** забирає з табличної бази даних усі формули підсумків.

Якщо цокнути на кнопці **OK**, Excel проаналізує базу даних і вставить формули, які були визначені, а також структурує таблицю.

РОЗДІЛ 3. ТЕХНОЛОГІЯ РОБОТИ З РЕЛЯЦІЙНИМИ БАЗАМИ ДАНИХ В СЕРЕДОВИЩІ СУБД ACCESS

Призначення MS Access

Microsoft Access є 32-розрядною системою управління реляційними базами даних, до складу яких входять таблиці, запити, форми, звіти, макроси та модулі, які зберігаються в одному файлі. Дані в такі бази даних можна вводити за допомогою спеціальних форм (шаблонів), зберігати їх у вигляді зв'язаних таблиць, вибирати з них потрібні відомості, обробляти, друкувати і т.ін. При наявності готової бази даних за допомогою Microsoft Access можна виконувати у ній пошук даних за критеріями, які задає користувач. Для створення готової БД за допомогою традиційних програмних продуктів розробнику необхідно виконати велику роботу зі створення окремих компонент, таких як форми, звіти та запити. Графічний інтерфейс Microsoft Access дозволяє за допомогою мишки зробити процес введення/виведення даних та зв'язування таблиць простим і наочним. Для автоматизації цього процесу у програму Microsoft Access було включено ряд спеціалізованих програм, які розв'язують такі задачі:

- **Конструктор/Design** надає у розпорядження користувача ряд інструментальних засобів для створення форм, запитів і звітів, а саме: кнопки, поля, ілюстрації і т.ін.
- **Майстер/Wizard** під час роботи задає ряд запитань користувачу та на основі відповідей будує закінчену форму або звіт.
- **Побудовник виразів/Expression Builder –** допомагає користувачеві при побудові виразів у таблицях, звітах, запитах, формах. Expression Builder містить список готових виразів і функцій.
- **Маска введення/Input Mask** робить можливим збільшення швидкості та точності введення. Маски введення визначають шаблони, яким повинні задовольняти дані, котрі вводяться у таблиці.

Етапи створення і роботи з базами даних

Для створення нової бази даних необхідно:

1. Запустити програму Microsoft Access 2016.
2. Вибрати пункт **Пуста настільна база даних/Blank desktop database**, і у діалоговому вікні, що з'явилося, зазначити ім'я файлу бази даних та його місцерозташування і цокнути на пункті **Створити/Create**. Головне вікно програми Microsoft Access є доступним.
3. Спроектувати **структуру бази даних,** визначивши поля з яких складатиметься окремий запис бази даних і задати тип кожного поля, а також призначити ім'я кожному полю.

4. Увести дані, використовуючи спеціальні **форми** (бланки). Це спрощує як введення, так і виведення окремих записів.
5. Пошук інформації у базі даних можна здійснювати на основі критеріїв пошуку, які застосовують до окремих полів бази даних. Користуючись критерієм Microsoft Access перебере всі записи бази даних і відбере лише ті, які задовільняють критерію. Сукупність критеріїв пошуку, призначених для фільтрації інформації, називають **запитом**.
6. Оформлення записів у **звіти** дозволяє представити дані у формі, зручній для подальшого використання.

Проектування, створення, введення даних та збереження таблиці бази даних

Після відкриття головного вікна програми, Microsoft Access пропонує формувати уміст об'єкта бази даних з назвою **Таблиця1/Table1**. Користувач має можливість або відразу вводити дані у комірки таблиці, або ж перейти в режим конструктора і сформувати структуру таблиці і після цього наповнювати таблицю даними. З самого початку активними є команди вкладки **Поля/Fields**. Перемикання між режимами таблиці/конструктора здійснюють послуговуючись командою **Подання/View** згаданої вище вкладки. В ході перемикання Access запропонує користувачеві задати назву таблиці, що відмінна від тієї, яка є за замовчуванням або погодитись з останньою. Під структурою таблиці розуміють: склад полів, їх імена, тип даних кожного поля, ключі та інші властивості полів.

Для визначення поля у стовпчиках **Ім'я поля/Field Name**, **Тип даних /Data Type, Опис/Description** задають відповідно його ім'я, тип даних і короткий коментар. У розділі **Властивості поля/Field Properties** на вкладці **Загальні/General** задають властавості поля: **Розмір поля/Field Size**, **Формат/Format**, **Підпис/Caption**, **Правило перевірки/Validation Rule, Текст перевірки/ValidationText** та ін. На вкладці **Підстановка/Lookup** вибирається тип елемента керування: **Текстове поле/Text Box, Список/List Box**, **Поле зі списком/Combo Box**.

Після визначення структури таблиці її потрібно зберегти за допомогою команди **File→Save/Файл→Зберегти** або кнопки **Зберегти/Save** панелі швидкого доступу.

Безпосереднє введения даних у таблицю здійснють в режимі таблиці. Перехід у режим таблиці з режиму конструктора виконують командою **Подання/View** вкладки **Основне/Home**.

Допустимі типи полів таблиць

У колонці **Ім'я поля/Fild Name** задається ім'я поля. Воно може мати довжину до 64-х символів, включати кирилицю, пропуски та спеціальні символи, окрім крапок, знаків оклику та дужок. Для кожного поля

обов'язково задають його тип, який визначає вид даних, які будуть зберігатись у даному полі. Microsoft Access розрізняє наступні типи полів:

- **Короткий текст/Short Text.** Короткі буквено-числові значення, наприклад прізвище або назва вулиці й номер будинку. Звертаємо увагу, що, починаючи з версії Access 2013, використовуваний раніше тип даних "Текст" тепер має називу «Короткий текст».
- **Довгий текст/Long Text.** Типовий приклад використання поля цього типу – детальний опис товару. Звертаємо увагу, що починаючи з версії Access 2013, використовуваний раніше тип даних «Memo» тепер має називу "Довгий текст".
- **Число/Number**. Числові поля містять будь-які числові значення. Діапазон допустимих значень визначається параметром **Розмір поля/Field Size**.
- **Дата й час/Date/Time**. Поля дати й часу містять значення дат і часу в діапазоні від 100 до 9999 року.
- **Грошова одиниця/Currency**. Грошові поля. У таких полях можна зберігати числа з точністю до 15-и розрядів зліва від коми і з точністю до 4-х десяткових розрядів (зазвичай 2-х) справа від десяткової коми.
- **Автонумерація/AutoNumber**. Поле лічильника містить число, яке автоматично збільшується на одиницю, коли в таблицю додають запис.
- **Так/Ні/Yes/No**. У таких полях зберігають значення Так (Yes) або Ні (No). Поля даного типу не індексуються.
- **Об'єкт OLE/OLE Object**. В OLE полях поміщають такі об'єкти, як таблиця MS Excel або Draw тепер має називу графіка і т.ін. Розмір поля може бути до 256 Mb. Поля даного типу не індексуються.
- **Гіперпосилання/Hyperlink**. Текст або поєднання тексту та чисел, що зберігаються в текстовому форматі й використовуються як адреса гіперпосилання.
- **Вкладення/Attachment.** Зображення, файли електронних таблиць, документи, діаграми та інші типи підтримуваних файлів, вкладені в записи бази даних (схожі на вкладені файли в електронних листах).
- **Обчислювальний/Calculated.** Результати обчислення. Обчислення мають містити посилання на інші поля в тій самій таблиці. Щоб створити обчислення, використовують побудовник виразів.
- **Майстер підстановок/Lookup Wizard.** Список значень, отриманих із таблиці або запиту, або набір значень, який вказують, коли створюють поле. Якщо вибрати цей тип даних, запуститься майстер підстановок, за допомогою якого можна створити поле підстановки. Поле підстановки підтримує тип даних "Число" або "Короткий текст", залежно від того, що вибрати в майстрі.

Визначення параметрів поля таблиці

Характеристики кожного поля визначають рядом параметрів, які регламентують способи обробки, збереження та ідентифікації даних.

Параметри поля перелічені в режимі проектування в нижній частині діалогового вікна **Властивості поля/Field Properties**:

- **Розмір поля/Field Size –** визначає максимальну довжину текстового поля або спосіб представлення чисел.
- **Формат/Format –** визначає спосіб представлення даних. Поряд з наперед визначеними стандартними форматами допускається використання власних форматів користувача.
- **Нові значення/New Values –** визначає нове значення для лічильника.
- **Кількість знаків після коми/Decimal Places –** визначає кількість розрядів праворуч від десяткової коми.
- **Маска вводу/Input Mask –** визначає при введенні формат даних, який включає відображення постійних символів у полі, а також задає перевірку формату даних.
- **Підпис/Caption –** визначає надпис, який буде використовуватися в якості імені поля у формі або звіті.
- **Значення за промовчанням/Default Value –** визначає значення, яке автоматично буде введене у поле при генерації запису.
- **Правило перевірки/Validation Rule –** правило яке обмежує допустимі для введення у поле дані.
- **Текст перевірки/ValidationText –** повідомлення про спробу ввести у поле дані, які не задовольняють правилу, яке задане характеристикою Правило перевірки/Validation Rule.
- **Обов'язково/Required –** визначає необхідність заповнення даного поля при введенні.
- **Індексовано/Indexed –** ознака індексування, яка набуває наступних значень: **Ні/No** при відсутності індексування, **Так (Повторення дозволені)/Yes (Duplicates OK)** індексування з можливістю повторення ключів та **Так (Без повторень)/Yes (No Duplicates)** без такої можливості.

Визначення розміру поля у таблиці

Розмір поля визначається діапазоном значень, які можна зберігати у полі. Розмір поля задають за допомогою параметра **Розмір поля/Field Size**. Для числових полів цей параметр може мати, зокрема, одне з наступних значень:

- **Байт/Byte –** у полі можуть зберігатися числа від 0 до 255 (цілі). Поле займає один байт.
- **Ціле число/Integer –** у полі можуть зберігатися числа від -32768 до 32767 (цілі). Поле займає два байти.
- **Довге ціле число/Long Integer –** у полі можуть зберігатися цілі числа від -147483648 до 2147483647 поле займає чотири байти.

- **Одинарне значення/Single –** у полі можна зберігати дробові числа зі шістьма знаками в дробовій частині в діапазоні від -3,402823 E38 до 3402823 E38.
- **Подвійне значення/Double –** у полі можна зберігати дробові числа з десятьма знаками у дробовій частині в діапазоні від – 1,79769313486232E38 до 1,79769313486232E38.

Визначення та задання ключового поля

Після того як були визначені всі поля, слід вибрати щонайменше одне поле як **первинний ключ**. Оголошення первинного ключа заборонить вводити однакові записи, оскільки поле таблиці, яке використовується в якості первинного ключа, містить унікальний ідентифікатор для кожного запису. Це поле не може мати однакову величину в різних записах. Первинний ключ може бути визначений лише в режимі проектування таблиці. Для цього необхідно виконати команду **Ключове поле/Primary Key** з вкладки **Конструктор/Design.** Марковане поле негайно помітиться піктограмою ключа в селекторній колонці (це і є ознака, що поле є первинним ключем) і відповідно індексується.

Створення обчислювального поля

Розглянемо, як створити обчислювальне поле у формі. Нехай на стрічці відображаються ескізи елементів керування. Цокають на піктограмі **Текстове поле/Text Box** для того, щоб у кінцевому випадку реалізувати обчислювальне поле. Після завершення розміщення поля його ім'ям буде **Текст0/Text0**, а текстове поле – вказівкою **Без прив'язки/Unbound**. Викликають вікно властивостей для об'єкта типу **Підпис/Label**. Параметр **Caption/Надпись** (Надпис) відповідає за назву обчислювального поля, ака відображатиметься для користувача поряд із самим текстовим полем. Відкривають вікно властивостей для елемента **Текстове поле/Text Box**. У полі **Джерело елемента керування/Control Source** вводять формулу обчислення. Будь-яка формула – це комбінація операторів та імен полів таблиці/запиту. Формула повинна починатися зі знаку „=", а ім'я поля, яке є в формулі необхідно брати у квадратні дужки ([…]).

При проведенні обчислень враховують таке:

- Для того щоб проводити обчислення коректно, ім'я поля у формі повинне точно співпадати з відповідним ім'ям поля у таблиці. Інакше з'явиться повідомлення про помилку (? Name).
- Microsoft Access «обчислює» у виразах лише ті поля, які були визначені при створенні таблиці і використовуються при проектуванні форми. Поле, додане заднім числом, не враховується, і тому не може бути використане в обчисленнях.
- Для того, щоб не набирати формулу безпосередньо на клавіатурі, можна скористатись засобом Побудовник виразів/Expression Builder.

Для його виклику натискують кнопку, розміщену праворуч від поля **Джерело елемента керування/Control Source**.

Зміна структури та задання параметрів відображення таблиці

Зміну структури таблиці здійснюють у режимі конструктора, хоча такі операції, як заміна імені, додавання, вилучення неключових полів, можна виконувати у режимі таблиці.

Параметри відображення таблиці на екрані змінюютъ послуговуючись групою команд **Форматування тексту/Text Formatting** з вкладки **Основне/Home** або діалоговим вікном **Форматування таблиці даних/Datasheet Formatting**. Це вікно з'являється, якщо цокнути на значку цієї групи. Операції зміни вигляду таблиці можна виконувати за допомогою мишки. Наприклад, для зміни ширини стовпця курсор мишки встановлюють на лінію, яка розділяє імена стовпців, після чого границю стовпця можна перемістити у потрібне місце. Для змінии висоти рядка курсор мишки встановлюєгься на границі між записами в області маркування записів, після чого границю рядка переміщають на погрібну відстань. При цьому змінюється висота всіх рядків таблиці.

Здійснення переміщення по таблиці

Покрокове переміщення від запису до запису за допомогою клавіш керування курсором у великих таблицях може зайняти багато часу. За допомогою команди **Перейти/Go To** групи команд **Пошук/Find** вкладки **Основне/Home** можна перейти до конкретного запису.

Перший запис/First – курсор переміщається з поля активного запису у таке ж поле першого запису.

Останній запис/Last – курсор переміщається з поля активного запису у таке ж поле останнього запису.

Наступний запис/Next – курсор переміщається з поля активного запису у таке ж поле наступного запису.

Попередній запис/Previous – курсор переміщається з поля активного запису у таке ж поле попереднього запису.

Створити/New – курсор встановиться у порожньому записі в кінці таблиці, після чого можна вводити новий запис.

Засоби пошуку та заміни даних у таблиці бази даних

Якщо розмір таблиці великий, то пошук потрібних даних за допомогою переміщення по таблиці буде проблематичним. Якщо уміст шуканого поля відомий, то його можна знайти з допомогою функції пошуку. Для цього необхідно виконати команду **Знайти/Find**, що міститься у групі **Пошук/Find** вкладки **Основне/Home**. З'являється діалогове вікно **Пошук і заміна/Find and Replace**: у полі **Find What/Знайти** задаємо об'єкт для

пошуку та в полі **Search/Шукати** вказуємо напрямок пошуку, наприклад, **Down/униз**.

Пошук запускається після натиснення кнопки **Знайти далі/Find Next**. Якщо об'єкт пошуку буде знайдений, то Microsoft Access виділить вміст поля, а в рядку статусу з'явиться повідомлення **Пошук виконано/Search succeeded**. За допомогою зазначеної вище кнопки цього діалогового вікна можна віднайти інші входження шуканого об'єкта.

Суттєво розширити коло пошуку можна, якщо застосувати символи-замінники „*" і „?". При цьому „*" заміняє будь-яку кількість символів, а "?" – лише один символ. Для заміни умісту поля вибирають закладку **Замінити/Replace** аналізованого діалогового вікна, а далі у полі введення **Find What/Знайти** вказують те, що замінити, а в полі **Замінити на/Replace With** – нові дані. Запускають процес заміни, натиснувши кнопку **Знайти далі/Find Next**. Такий спосіб запуску заміни призведе до того, що Microsoft Access знайде та позначить об'єкт, що підлягає заміні, але заміни не виконає доти, поки не вирішить сам користувач. Для автоматичної зміни без запитань необхідно натиснути кнопку **Замінити все/Replace All**.

Типи зв'язків між таблицями бази даних

Microsoft Access дозволяє створювати реляційні бази даних. В реляційній базі даних користувач може визначити зв'язки (відношення) між декількома таблицями. Microsoft Access враховує ці відношення при пошуку взаємозв'язаних даних та при обробці запитів, форм і звітів, які ґрунтуються на декількох таблицях. Між двома таблицями можна встановити зв'язок-об'єднання за деяким полем зв'язку. Для цього можна вибрати один із таких способів об'єднання записів:

- об'єднання тільки тих записів, у яких зв'язані поля обох таблиць збігаються;
- об'єднання тих записів, у яких зв'язані поля обох таблиць збігаютъся, а також об'єднання усіх записів із першої таблиці, для яких немає зв'язаних у другій, із порожнім записом другої таблиці;
- об'єднання тих записів, у яких зв'язані поля обох таблиць збігаються, а також об'єднання усіх записів із другої таблиці, для яких немає зв'язаних у першій, із порожнім записом першої таблиці.

Створення схеми даних починається в області переходів виконанням команди **Зв'язки/Relationships** вкладки **Знаряддя бази даних/Database Tools**. У вікні **Відображення таблиці/Show Table** вибирають таблиці та запити, які включають у схему даних. Вибір виділеного об'єкта здійснюють за допомогою кнопки **Додати/Add**.

Процедура створення зв'язків між таблицями бази даних

Для встановлення зв'язків між таблицями необхідно виконати команду **Зв'язки/Relationships** вкладки **Знаряддя бази даних/Database Tools**.

Відкриється вікно, у якому зображаються таблиці бази даних із вказаними зв'язками (якщо вони були раніше визначені, якщо ні, – то вікно порожнє).

Для додавання таблиці до схеми даних необхідно вибрати команду **Відобразити таблицю/Show Table** вкладки **Конструктор/Design**. На екрані з'явиться вікно **Відображення таблиці/Show Table**, у якому можна вибрати потрібні таблиці і/або запити та натиснути кнопку **Додати/Add** і закрити вікно за допомогою кнопки **Close/Закрити**.

Для створення зв'язку за допомогою мишки перетягують поле, яке слід використовувати для зв'язку (зазвичай ключове поле), зі списку головної таблиці на відповідне поле підпорядкованої таблиці. У діалоговому вікні **Редагування зв'язків/Edit Relationships**, яке з'явиться на екрані, необхідно вибрати параметри зв'язку та натиснути кнопку **Створити/Create**. При цьому забезпечується відношення **Один-до-багатьох/One-To-Many**, що відображається в області **Тип зв'язку/Relationship Type**. Це означає, що одному запису головної таблиці буде поставлено у відповідність декілька записів із підлеглої, але кожний запис із підлеглої таблиці буде зв'язаний лише з одним записом головної таблиці. Таке відношення є найпоширеніше у реляційних базах даних.

Дві опції **Каскадне оновлення пов'язаних полів/Cascade Update Related Fields** і **Каскадне видалення пов'язаних записів/Cascade Deleted Related Records**, які доступні лише при увімкненому індикаторі **Забезпечення цілісності даних/Enforce Referential Integrity**, забезпечують каскадне оновлення полів і каскадне знищення пов'язаних запиів для зв'язаних таблиць.

Засоби створення форм MS Access

Для комфортнішої роботи з базами даних використовують форми. Форма служить гарантією збереження бази даних у потрібному вигляді. Крім цього, вона може відгороджувати конфіденційну інформацію від дій некваліфікованого користувача. В Microsoft Access є, зокрема, такі засоби для створення форм:

- **Форма/Form –** автоматично створює форму, яка будується на вибраній таблиці (запиті), на основі стандартної форми. Це найпростіший спосіб створення форми.
- **Майстер форм/Form Wizard** – автоматично створює форму на основі вибраних полів таблиці та запитів і пропонує на вибір одну із стандартних форм і стилі її оформлення.
- **Конструктор форм/Form Design** – створює порожній бланк (макет), у якому користувач за допомогою інструментальних засобів (група команд з назвою **Елементи керування/Controls**) може створити власну форму.
- **Пуста форма/Blank Form** – створює форму без елементів управління та формату.

– **Навігація/Navigation** – створює форму, яка дає змогу користувачам переходити до інших форм і звітів.
– **Додаткові форми/More Forms** – дозволяє вибрати можливість побудови форм типу: «**Кілька елементів/Multiple Items**», «**Таблиця/Datasheet**», «**Розділена форма/Split Form**», «**Модальне діалогове вікно/Modal Dialog**»

Форми можна створювати як на основі таблиць, так і на основі запитів. Якщо форма збережена, то при кожному її викликові запит активізується заново.

Для створення форми необхідно скористатися потрібною командою з групи **Форми/Forms** вкладки **Створення/Create**.

Види форм в MS Access

- **Форма одного елемента (Form)** – для кожного запису використовують одну сторінку форми. Застосовується для записів з великою кількістю полів.
- **Розділена форма (Split Form)** – дає змогу одночасно переглядати дані у двох поданнях: у поданні форми та у вікні табличного подання даних. Робота з розділеними формами дає змогу користуватися перевагами двох типів форм в одній формі. Наприклад, можна використовувати вікно табличного подання даних для швидкого пошуку запису, а потім подання форми для перегляду й редагування цього запису. Ці два подання зв'язані з одним джерелом даних і постійно синхронізуються між собою.
- **Форма, у якій відображається кілька записів (Multiple Items)** – аорму з кількома елементами, яку також називають неперервною формою, зручно використовувати, якщо необхідно, щоб у формі відображалося кілька записів, але в ній було більше можливостей для настроювання, ніж у даних у табличному поданні,.
- **Таблиця (Datasheet)** – виводиться у вигляді таблиці та використовується як підпорядкована форма в складених формах.
- **Форма, яка містить підформи (One-to-many form)** – створюється форма, яка складається з двох форм – головної та підпорядкованої, які зв'язані між собою зв'язком One-To-Many.
- **Форма навігації (Navigation form)** – це звичайна форма, яка містить навігаційний елемент керування. Форми навігації – це чудове доповнення для будь-якої бази даних, але якщо базу даних планується публікувати в Інтернеті, створення форми навігації відіграє особливо важливу роль, оскільки у браузері область переходів Access не відображається.

Процедура створення форми за допомогою конструктора

Створити порожню форму та далі привнести потрібні поля, елементи оформлення і керуючі елементи можна так:

1. знаходячись на вкладці **Створення/Create** із групи **Форми/Forms** вибирають команду **Конструктор форм/Form Design**. З'являється вікно **Подробиці/Detail**, на якому формують уміст форми.
2. У групі **Знаряддя/Tools** вибирають команду **Додавання наявних полів/Add Existing Fields**. З'являється вікно Список полів/FieldList. Цокаємо на посиланні **Відображати всі таблиці/Show all tables**. Перетягуємо потрібні поля в область **Подробиці/Detail**.
3. При потребі звертаємося до властивостей об'єктів форми. Для цього виділяємо мишкою цей об'єкт і зі згаданої вище групи вибираємо команду **Аркуш властивостей/Property Sheet**.

Режим конструктора форм надає, зокрема, такий комплекс засобів для створення форм:

1. **Конструктор/Design View –** вікно проектування форми.
2. **Елементи керування/Controls** – група команд для монтажу елементів керування та зображення.
3. **Теми/Themes** – група команд для зміни загального вигляду бази даних, кольорів та шрифтів.
4. **Колонтитули/Header / Footer** – група команд для монтажу емблеми, заголовка, дати та часу.
5. **Знаряддя/Tools** – група команд для додавання полів таблиць, для доступу до властивостей об'єктів форми, а також зміни послідовності переходу для елементів керування. Є можливість відобразити тексти процедур для форми.

Створення форми за допомогою майстра

Створення екранної форми за допомогою **Майстер форм/Form Wizard** складається з декількох основних етапів:

- Перш за все необхідно вибрати таблицю або запит бази даних, на основі якої (якого) буде створена форма.
- Для поодинокого включення поля потрібно цокнути на кнопці >, внаслідок чого поле бази даних включається до форми. За допомогою кнопки >> одночасно можна включити усі поля. Кнопка <, а також << дають можливість виключити відповідно одне або усі поля бази даних. Після вибору полів запиту слід натиснути кнопку **<Далі >>/<Next >>** і перейти до наступного кроку.
- Далі обирають макет для форми. Access пропонує такі макети: **Стовпці/Columnar**, **Таблиця/Tabular**, **Таблиця даних/Datasheet**, **За шириною/Justified**.
- На останньому етапі задають назву форми. Далі майстер пропонує два варіанти: відкрити форму для перегляду або введення даних, і другий варіант – запустити режим редагування екранної форми для дооснащення її додатковими елементами. Сам по собі майстер створює розумний мінімум елементів управління.

Якщо необхідно відредагувати екранну форму або привнести у неї обчислювальні поля, то вмикають режим конструктора екранних форм (команда **Конструктор/Design View**, яка міститься у групі **Подання/View** вкладки **Основне/Home**). Послуговуючись інструментами конструктора, можна модифікувати вже існуючі поля або започаткувати нові. Для того, щоб додати поле обчислень, потрібно у наборі елементів керування вибрати відповідну команду. Далі перетягнути її мишкою у потрібне місце форми. Елемент управління **Підпис/Label** перейменувати, а по полю **Unbound** (назва за замовчуванням) цокнути правою клавішею мишки та вибравши команду **Властивості→Дані/Properties→Data** вписати формулу для обчислення у полі **Джерело елемента керування/Control Source**. Для запису формули можна використати **Побудовник виразів/Expression Builder**. Зміни зроблені в режимі конструктора треба зберегти. Результат редагування переглядають у режимі форми. При потребі екранну форму можна роздрукувати.

Доповнення форми елементами управління

При створенні екранних форм у режимі редагування чи за допомогою майстра вони матимуть мінімальний набір елементів управління. Це можуть бути кнопки, які забезпечують перехід від одного запису до іншого, на початок чи на кінець бази даних. Однак практика використання екранних форм вимагає суттєвого розширення їх функціональних можливостей. Екранні форми можуть бути оснащені радіокнопками, клавішами управління та іншими елементами. Елементи управління в екранну форму можна привнести двома способами:

- Традиційний – це використання ручного режиму редагування, по-іншому **конструктора форм**. Але від користувача у цьому випадку вимагається досить глибоке розуміння основ використання об'єктів та візуального стилю програмування. Запускається редагування екранної форми командою **Конструктор/Design View**, яка міститься у групі **Подання/View** вкладки **Основне/Home**. А відтак, на екрані додатково з'являється вкладка **Конструктор/Design**, на якій вибирається потрібний інструмент-команда.
- Існує інший, спрощений варіант створення елементів управління. Він ґрунтується на послуговуванні майстром. У цьому випадку процес привнесення елементів управління в екранну форму складається з таких етапів:

1. Відкривають у режимі конструктора деяку екранну форму. У групі **Елементи керування/Controls** вкладки **Конструктор/Design** в розділі **Додатково/More** вибирають команду **Застосувати майстри елементів керування/Use Control Wizards**. Нехай для прикладу розглянемо монтаж елемента управління «Кнопка». У діалоговому вікна треба зазначити, яку дію слід виконувати в разі натискання кнопки. Візьмемо для прикладу, **Операції з записами/Record Operations**. Вибирають **<Далі >>/<Next >>** та переходять до наступного етапу.

2. На цьому етапі користувач вказує, що використовувати на кнопці: текст чи зображення. Цокають **<Далі >>/<Next >>**.
3. У наступному вікні майстра вказують ім'я кнопки задля того, щоб згодом посилатися на кнопку. На завершення роботи майстра цокають **<Готово>/<Finish>**. Відтак, зазначений вище елемент керування вмонтований.

Відфільтровування інформації задля відображення у формі

Фільтр служить лише для селекції, сортування записів. В результаті фільтрації або сортування у формі будуть ідентифікуватися лише ті дані, які задовільняють критерій (або відсортовані). До форм застосовують три способи фільтрації даних:

- **Фільтр за формою** – фільтрація даних у формі за допомогою простих умов. Для активізації фільтра виконують команду **Додатково▼/Advanced▼** групи **Сортування й фільтр/Sort & Filter** з вкладки **Основне/Home** і вибирають підкоманду **Змінити фільтр/ Filter By Form**. У діалоговому вікні **Фільтрування за формою/Filter by Form** переходять на закладку **Шукати/Look For** і вказують умови вибору у відповідному полі. Для задання складніших умов відбору записів послуговуються закладками **Або/Or**. Врешті-решт виконують команду **Застосувати фільтр/Toggle Filter** або цокають правою кнопкою мишки в аналізованому діалоговому вікні і вибирають **Застосувати фільтр/сортування/Apply Filter/Sort**.
- **Фільтр за виділеним –** фільтрація даних у формі за допомогою виділення даних. Насамперед виділяють дані, які служитимуть критерієм фільтрування або розміщують курсор на значенні поля. Далі виконують команду **Виділення▼/Selection▼** групи **Сортування й фільтр/Sort & Filter** вкладки **Основне/Home**. Із меню підкоманд вибирають потрібну. Після застосування фільтрування у нижній частині форми з'явиться відфільтровані записи.
- **Розширений фільтр/сортування –** розширена побудова фільтру для форми. Для використання виконують команду **Додатково▼/Advanced▼** з вкладки **Основне/Home**, а далі підкоманду **Розширений фільтр/сортування…/Advanced Filter/Sort…** . З'являється вікно, яке нагадує вікно запиту. Перетягують мишкою ім'я поля, по якому відбуватиметься фільтрування, в QBE-область. У полі **Сортування/Sort** встановлюють, наприклад, опцію **За зростанням/Ascending** для сортування по зростанню. У полі **Критерії/Criteria** вводять об'єкт фільтрування. На вкладці **Основне/Home** натискають кнопку **Застосувати фільтр/ Toggle Filter**.

Створення головної та підпорядкованої форми

Майстер форм (а також конструктор форм) дає можливість створення досить складних конструкцій із декількома пов'язаними таблицями, які

називаються складеними формами. У цьому випадку використовуються дві таблиці, пов'язані відношенням **Один-До-Багатьох/One-To-Many**. У головній формі відображається зміст деякого запису головної таблиці, а в підпорядкованій формі з'являються залежні записи зв'язаної таблиці. При цьому підпорядкована форма вбудовується у головну форму таким чином, що вони обидві відображаються в одному вікні.

При побудові складеної форми за допомогою майстра необхідно виконати наступну послідовність дій:

- Запустити майстер форм – команда **Майстер форм/Form Wizard** з групи **Форми/Forms** вкладки **Створення/Create**.
- У першому діалоговому вікні з переліку **Таблиці та запити/Tables/Queries** вибирають таблицю чи запит, на основі якої будуватиметься форма.
- Формують список **Вибрані поля/Selected Fields**.
- Далі у списку таблиць/запитів вибирають підпорядковану таблицю та її поля.
- У наступному діалоговому вікні встановлюють конструкцію форми. Наприклад, **Форма з підформою (підформами)/Form with Subforms(s)**.
- Вибирають макет оформлення підформи.
- На останньому кроці задають імена форми та підформи і натискують кнопку **<Готово>/<Finish>**.

Додання заголовків, зауважень та колонтитулів у форму

Існують два різновиди верхніх і нижніх колонтитулів стосовно форми: **Верхній/Нижній колонтитул сторінки (Page Header/Footer)** і загальні: **Верхній/Нижній колонтитул форми (Form Header/Footer)**. Вони з'являються на верхніх і нижніх краях вікна форми і при необхідності їх можна додавати чи видаляти попарно. „Шапку" форми можна використовувати для відображення додаткової інформації, такої як заголовок або дата. Зауваження форми використовуються для інструкцій із заповнення, подання загальної суми.

Для подання заголовка та зауваження форми у режимі проектування правою кнопкою мишки цокають в області макета і у контекстно-залежному меню вибирають команду **Заголовок/примітка форми (Form Header/Footer)**. На екрані у вікні проектування з'являться зони заголовка (Верхній колонтитул форми/Form Header) і зауваження (Нижній колонтитул форми/Form Footer). У цих зонах можна розмістити елементи управління таким самим чином, як це робиться в основній зоні форми.

Найуживаніші керуючі елементи

- **Вибір/Select** – якщо активізована ця команда, то курсором мишки можна помічати поля, виконувати перенесення і змінювати розміри керуючих елементів.

- **Підпис/Label** – вставка у форму напису.
- **Текстове поле/Text Box** – відображення умісту активного запису поля бази даних або обчислювального поля.
- **Група параметрів/Option Group** – створення і розміщення групи, в яку можна ввести контрольні індикатори, селекторні кнопки або двопозиційні кнопки.
- **Перемикач/Toggle Button** – відображення значення деякої опції, яка може мати два стани.
- **Перемикач/Option Button** – вибір (індикація) одного з декількох взаємовиключних значень деякого параметра.
- **Прапорець/Check Box** – індикація (зміна) значення опції, яка може мати одне з двох можливих значень (увімкнено/вимкнено).
- **Поле зі списком/Combo Box –** після клацання мишкою на маніпуляторі відкриття (кнопка зі стрілкою з правого боку поля) відкривається комбінований список.
- **Список/List Box** – у полі списку, на відміну від комбінованого списку, всі елементи постійно видно. Якщо їх багато, так що вони не поміщаються у відведеному місці, то на правому краю поля з'являється лінійка прокрутки списку.
- **Кнопка/Button –** створення кнопки, яка може бути пов'язана з деякою командою або послідовністю дій.
- **Вставити зображення▼/Insert Image▼** – монтаж статичних ілюстрацій (графічних файлів) у форму.
- **Вільна рамка об'єкта/Unbound Object Frame –** монтаж у форму об'єкта, який знаходиться в окремому файлі ззовні бази даних.
- **Приєднана рамка об'єкта/Bound Object Frame** – вбудова ілюстрації або будь-якого іншого OLE-об'єкта, який зберігається у таблиці бази даних Microsoft Access.
- **Вставити розрив сторінки/Insert Page Break** – повідомлення для Microsoft Access, де повинен бути розрив форми на сторінці.
- **Підформа/підзвіт/Subform/Subreport** – вбудова підпорядкованої форми в головну форму та визначення відношень між формами.
- **Лінія/Line** – за допомогою даної піктограми в формі можна провести пряму лінію.
- **Прямокутник/Rectangle** – об'єднання та виділення груп полів, включаючи їх у прямокутну рамку.

Створення запиту до бази даних

За допомогою запитів користувач може вибрати з таблиці необхідні дані та вивести їх на екран. В ході підготовки запиту реалізується спеціальний фільтр, який пропускає лише потрібні користувачеві записи. Для створення запиту в режимі конструктора необхідно:

– Виконати команду **Конструктор запитів/Query Design** з групи **Запити/Queries** вкладки **Створення/Create**.

– З'являється діалогове вікно **Відображення таблиці/Show Table**, у якому вибирають таблицю (таблиці) і/або запит (запити), на основі яких будуватиметься запит.
– У списку полів зазначають поля, які використовуватимуться у запиті.
– У рядку **Сортування/Sort** у потрібному стовбці відкривають список опцій і вибирають необхідну (зокрема, **За зростанням/Ascending** – сортування за зростанням, **За спаданням/Descending** – сортування за спаданням).
– Закривають вікно запиту та, зазначивши ім'я, зберігають його.

Підготовка запиту для виокремлення з бази даних лише частини інформації завершена. Збережений запит буде відображатися в області переходів у розділі **Запити/Queries**.

Запит запускають на виконання командою **Запуск/Run** з групи **Результати/Results** вкладки **Конструктор/Design**. В результаті відображатимуться лише ті дані, які визначені запитом. Але відображення деякого поля у запиті можна заблокувати в режимі конструювання запиту. Для цього вибирають опцію **Відображення/Show** – в результаті на екрані відображатимуться лише ті поля, для яких встановлена ця опція.

Типи запитів

Поряд із запитами на вибірку за допомогою Microsoft Access можна реалізувати запити дій. Послуговуючись запитом дії користувач може змінювати або переносити дані в таблицях, а також активізовувати, додавати, видаляти групи записів, створювати нові таблиці. Розрізняють такі типи запитів дій:

– **Запит на додавання –** записи однієї таблиці (всі або лише відібрані) можна додати у кінець іншої таблиці.
– **Запит на видалення –** користувач може забрати групу записів, які відібрані за певним критерієм.
– **Запит на оновлення** – користувач може змінити групу записів, відібраних на основі критерію.
– **Запит на створення таблиці** – з динамічного набору записів, відібраних в результаті запиту, можна створити таблицю.

Способи створення запитів

Проектування запиту можливе лише тоді, коли існує відповідна база даних. Для того, щоб згенерувати запит існує дві можливості. Перша – запити за зразком (QBE – Query By Example), для яких потрібно вказати параметри запиту у вікні конструювання, задаючи зразки для пошуку інформації; друга – сконструйовані запити, коли користувач повинен за допомогою спеціальної мови запитів описати свій запит, використовуючи команди та функції мови SQL (Structured Query Language).

Для початківців рекомендують послуговуватися запитами за зразком. Можна сформувати запит самостійно за допомогою конструктора (команда

Конструктор запитів/Query Design) або скористатися майстром запитів (команда **Майстер запитів/Query Wizard**).

При першому вивченні доцільно скористатись майстром запитів.

Майстер запитів: різні варіанти запиту на вибірку

У найзагальнішому випадку, для того, щоб створити запит послуговуються або майстром, або конструктором запитів. Для цього вибирають одноіменні команди, а саме, **Майстер запитів/Query Wizard** або **Конструктор запитів/Query Design** з групи **Запити/Queries** вкладки **Створення/Create**. Розглянемо, які можливості щодо типів запитів надає майстер. Маємо:

- **Майстер простих запитів/Simple Query Wizard** дає можливість розпочати процес створення простого запиту за допомогою майстра запитів.
- **Майстер перехресних запитів/Crosstab Query Wizard** – перехресний запит подає дані у вигляді таблиці.
- **Майстер пошуку повторюваних записів/Find Duplicates Query Wizard** – при виконанні даного запиту відбувається пошук записів даних, які повторюються у таблиці. Такі запити допомагають при пошуку записів, які частково дублюють один одного. Часто назва тієї самої фірми у різних каталогах написана по-різному. Запити цього типу допомагають виправити аналізовану помилку.
- **Майстер пошуку незв'язаних записів/Find Unmatched Query Wizard –** дозволяє робити пошук записів даних у головній таблиці, для яких немає записів у підлеглих таблицях. Це можуть бути товари, на які немає попиту, клієнти, які нічого не замовляють і т. ін.

Використання Майстра запитів

Розглянемо, як будувати найпростіші запити за допомогою майстра. Для цього на вкладці **Створення/Create** у групі **Запити/Queries** вибирають команду **Майстер запитів/Query Wizard**. В результаті з'являється діалогове вікно, за допомогою якого необхідно:

- Визначитися із типом майстра. У нашому випадку це **Майстер простих запитів/Simple Query Wizard**.
- Далі вибирають базу даних, для якої буде створений запит.
- Для поодинокого включення поля потрібно цокнути на кнопці >, внаслідок чого розглядуване поле бази даних буде додане у запит. За допомогою кнопки >> одночасно всі поля можна включити у запит. Кнопка <, а також << дають можливість виключити, відповідно, одне або всі поля бази даних із запиту.
- Після вибору полів запиту слід цокнути на кнопці **<Далі >>/<Next >>** та перейти до наступного кроку. Пройшовши усі кроки, можна створити запит зі присвоєним іменем.

Коли потрібно створити запит за певним критерієм, то використовуємо конструктор запиту. Відкривається діалогове вікно, де вказані поля, імена таблиць даних, критерії сортування. Вибираємо, у яких полях і за якими критеріями генеруватиметься запит. Зміни у режимі конструктора записуємо та запускаємо запит на виконання. При потребі запит можна роздрукувати.

Задання критерію пошуку для запиту

У межах запиту можна визначити критерії для відбору інформації. Для цього необхідно:

- Відкрити запит у режимі проектування.
- У рядку **Критерії/Criteria** увести критерій для поля (або полів).
- Виконати запит, перейшовши в режим виконання запиту.
- Символи замінники „*" і „?" працюють як і раніше („*" – заміняє декілька символів, „?" – один символ).

Формування критерію пошуку за допомогою конструктора виразів

За допомогою структурованої мови запитів SQL – Structured Query Language в Microsoft Access користувач може сформувати дуже складні за структурою критерії і обчислення у запиті. SQL-запит є послідовністю інструкцій, у які включаються вирази.

Для того, щоб на екрані відобразити SQL-запит або внести у нього зміни, необхідно у режимі проектування запиту цокнути правою кнопкою мишки на закладці з назвою запиту основної частини вікна програми Microsoft Access і вибрати команду **Режим SQL/SQL View**. Основою для створення запитів в мові SQL є інструкція **SELECT**. Ця інструкція визначає поля, які будуть входити у запит. При виконанні запиту з таблиць, які задані параметром **FROM**, фільтруються записи, які задовільняють умові **WHERE**. З відібраних записів виділяються тільки ті поля, які перелічені в інструкції **SELECT**. При роботі з інструкцією **SELECT** необхідно брати до уваги наступні зауваження:

- Якщо задавати більше одного поля, то імена слід розділяти комами. Перелік полів потрібно наводити у тій послідовності, в якій вони повинні бути відображені у запиті.
- Якщо використовують імена полів, які містять пропуски, тоді ім'я необхідно помістити у квадратні дужки.
- Якщо обробляються декілька таблиць одночасно, то у списку полів рекомендується давати повне ім'я, тобто Ім'я_таблиці та Ім'я_поля
- Параметр **FROM –** задає перелік таблиць або запитів, які містять потрібні поля. Використовуємо символ „*****", якщо необхідно задати всі поля таблиці.
- Параметр **WHERE** – необов'язковий. За допомогою цього параметра користувач визначає, які записи з таблиць з'являться у запиті. Microsoft Access вибирає записи, які задовільняють умови, визначені у **WHERE**.

- Параметр **IN** використовують при роботі з базами даних іншого (не Access) формату.

Способи створення звітів

Звітом називають сукупність полів БД, доповнених результатами обчислень і засобами унаочнення, поданими у наочній формі, зручній для сприйняття. Звіт найчастіше створюється у вигляді електронного документа, який після вивчення та остаточної редакції друкують на папері.

Створити звіт можна тоді, і лише тоді, коли вже існує відповідна база даних. Для одержання звіту існує декілька способів. Послідовно зупинимося на усіх цих способах. Команди для підготовки звіту містяться в групі **Звіти/Reports** вкладки **Створення/Create**. Зокрема, команда:

- **Звіт/Report** служить для створення стандартного звіту на основі даних поточного запиту або таблиці, до якого можна додати компоненти, наприклад групи або підсумки.
- **Майстер звітів/Report Wizard** дозволяє запустити майстер (wizard) звітів, за допомогою якого вся робота з створення звітів зводиться до виконання декількох нескладних етапів.
- **Етикетки/Labels** використовується для створення стандартних поштових наліпок на конверти під час розсилання серійних листів.
- **Конструктор звітів/Report Desing** дає змогу створювати новий порожній звіт у режимі конструктора. У цьому режимі можна вносити додаткові зміни до конструкції звіту, наприклад, додавати настроювані типи елементів керування та писати код.
- **Пустий звіт/Blank Report** потрібна для створення нового порожнього звіту, у який можна вставляти поля й елементи керування, а також інші засоби оформлення.

Створення нового звіту

1. На вкладці **Створення/Create** у групі **Звіти/Reports** вибрати одну із команд: **Звіт/Report, Майстер звітів/Report Wizard, Етикетки/Labels, Конструктор звітів/Report Desing** або **Пустий звіт/Blank Report**.
2. У випадку послуговування майстром переміщують потрібні поля у список **Вибрані поля/Selected Fields**.
3. У наступному вікні майстра звітів потрібно зазначити, за якими полями відбуватиметься групування. Дані в звіті можна групувати тільки за чотирма полями.
4. Чергове вікно служить для задання порядку сортування та відомостей зведення.
5. Поточне вікно майстра дозволяє вибрати макет для звіту та задати орієнтацію: книжкову або альбомну. Варіанти макетів такі: **Східчастий/Columnar**, **Блок/Tabular** і **Структура/Justified**.

Невеликий фрагмент звіту буде представлений у лівій частині діалового вікна.

6. На завершення майстру звітів необхідно вказати назву звіту і вибрати одну з опцій: **Переглянути звіт/Preview the report** чи **Змінити макет звіту/Modify the report's design**.

Створення звіту за допомогою майстра

Для створення звіту за допомогою майстра (wizard) необхідно на вкладці **Створення/Create** вибрати команду **Майстер звітів/Report Wizard**. В результаті з'являється однойменне діалогове вікно, яке дозволяє у режимі перегляду вибрати таблиці та запити, для яких буде створюватись звіт. Далі необхідно вказати ті поля бази даних, котрі увійдуть до створюваного звіту. Для поодинокого вибору треба натиснути на кнопку >, внаслідок чого помічене поле буде включене до звіту. А от за допомогою кнопки >> одночасно всі поля можна включити у звіт. Кнопка <, а також << дають можливість виключити одне або усі поля бази даних зі звіту. Наступні кнопки мають таке призначення:

- **Скасувати/Cancel –** дозволяє припинити процес створення звіту.
- **< Назад/< Back –** дає можливість повернутись на один крок назад.
- **Далі >/Next > –** перехід до наступного етапу одержання звіту.
- **Готово/Finish –** після виконання цієї команди усі проміжні етапи виконуються автоматично та одержується звіт.

Коли потрібно зробити якісь зміни у звіті: змінити шрифти, заголовки або ввести у звіт обчислювальне поле, то використовується **Конструктор звітів/Report Desing**. Саме тут виконують усі потрібні редагування уже готового звіту. Наприклад, для зміни шрифту у всьому звіті або в окремих його полях, потрібно відмітити усі поля (комбінація клавіш <Ctrl+A>) або конкретне поле. Далі послуговуються командою **Шрифти/Fonts** з групи **Теми/Themes** вкладки **Конструктор/Design**.

Для того, щоб доповнити звіт обчислювальним полем, потрібно скористатися елементом керування **Текстове поле/Text Box**. З'являються два об'єкти: один типу **Підпис/Label** для того, щоб відобразити текст-пояснення про обчислювальне поле, а інший **– Текстове поле/Text Box –** для реалізації формули обчислення. Відкривають вікно властивостей для останнього. У полі **Джерело елемента керування/Control Source** записують формулу для обчислення. Переходять у режим перегляду звіту. При потребі звіт можна роздрукувати.

ПРИКЛАДИ РОЗВ'ЯЗУВАННЯ ЗАДАЧ СТВОРЕННЯ І ПІДТРИМКИ ТИПОВИХ БАЗ ДАНИХ

Задача 1:

Імпортувати з MS Excel у MS Access таблицю зі списком студентів групи. Створити таблиці для оцінок кожного студента та списку викладачів; передбачити форми для перегляду та редагування даних таблиць. Для створеної БД:

- визначити середній бал по групі за кожним із предметів і середній бал по групі загалом;
- визначити середній бал по групі серед тих, хто має найвищу оцінку з вищої математики;
- створити звіт по всіх студентах; групування провести за першою літерою прізвища студента, а сортування всередині групи провести за оцінкою з філософії;
- створити поштові наліпки для усіх студентів групи;
- підготувати привітальні листи всім студентам, роздрукувавши їх оцінки, середній бал і рейтинг (середній бал відносно середнього по групі, помножений на 100);
- побудувати гістограму розподілу оцінок із вибраної користувачем дисципліни;
- створити меню для зручної роботи користувача у вигляді форми, яка має завантажуватися автоматично при відкритті БД.

Розв'язання задачі 1:

1. *Імпортуємо з MS Excel у MS Access таблицю зі списком студентів групи*[4].

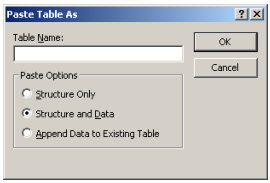
Після запуску MS Access з'являється область задач **New File**. У групі опцій **New** вибираємо пункт **Blank Database**. Далі Access пропонує зберегти БД. Вибираємо потрібну папку та вводимо назву БД «Навчання_Студентів». Імпортування даних з Excel здійснюється командою **File→Get External Data→Import…** . У діалоговому вікні **Import** вказуємо файл Excel, котрий містить список студентів групи. Далі в інтерактивному режимі покроково виконуємо вказівки майстра імпортування. В результаті перший об'єкт БД – таблиця «Відомості_Про_Студентів» є готовою.

2. *Створюємо таблицю для оцінок кожного студента.*

Скопіюємо таблицю «Відомості_Про_Студентів» у таблицю з іменем «Оцінки_Студентів» із збереженнм структури та даних першої. Для цього,

[4] Нехай потрібна таблиця підготовлена в Excel.

знаходячись у контейнері БД виділяємо об'єкт – таблицю «Відомості_Про_Студентів» – і цокаємо правою кнопкою мишки на виділеній назві таблиці та у контекстно-залежному меню вибираємо пункт **Copy**. Далі цокаємо правою кнопкою мишки на вільному місці панелі, яка відображає список об'єктів БД, та у меню, яке з'являється вибираємо пункт **Paste**. З'являється діалогове вікно **Paste Table As**, у якому нас цікавить опція **Structure and Data** і у цьому ж вікні задаємо назву нової таблиці – «Оцінки_Студентів».



Відкриваємо копію і змінюємо назви та значення останніх чотирьох полів таблиці. Нехай зазначені поля мають такі назви: Вища математика, Інформатика, Філософія, Історія економічних вчень. Зміна назви поля відбувається шляхом перейменування існуючих назв, після подвійного цокання на старій назві. Для зміни значень полів необхідно увійти у структуру таблиці та в розділі **Data Type** для кожного з останніх чотирьох полів встановити тип **Number**.

Переходимо із вікна конструктора (**Design View**) у режим таблиці (**Datasheet View**) та заповнюємо значеннями потрібні поля. Зберігаємо останній варіант побудованої таблиці «Оцінки_Студентів».

Оцінки_Студентів : Table

ID	ПІП студента	Вища математика	Інформатика	Філософія	Історія економічних вчень
1	Агащук Любов Федорівна	5	4	5	3
2	Березанський Орест Павлович	5	3	5	4
3	Войтович Іван Михайлович	4	4	5	3
4	Допілко Максим Петрович	4	4	5	4
5	Зелена Марта Євгенівна	5	4	4	4
6	Зозуля Олена Василівна	3	3	3	5
7	Кипибіда Марія Леонтівна	3	2	5	5
8	Красуся Ганна Степанівна	4	5	5	4
9	Макогін Оксана Орестівна	4	4	3	3
10	Макух Ганна Данилівна	4	5	5	3
11	Матюшевський Зеновій Якович	4	5	4	4
12	Мацьків Степан Степанович	5	4	5	4
13	Миколайчук Евгеній Прокопович	3	3	5	5
14	Миколишин Григорій Климович	4	4	5	5
15	Міронов Семен Петрович	4	3	4	5
16	Мішко Катерина Іванівна	3	5	4	5
17	Музичишин Артем Васильович	3	3	4	3
18	Назаркович Петро Богданович	5	4	3	5
19	Наконечний Роман Михайлович	5	4	3	5
20	Петльована Аліна Йосипівна	5	5	3	5
21	Попик Арсеній Федорович	3	5	4	4
22	Семенків Іван Ярославович	4	5	5	4
23	Семенчук Богдан Іванович	2	3	5	3
24	Собчук Катерина Львівна	3	4	3	3
25	Чередніченко Василь Степанович	5	4	4	3
26	Яцків Леонід Романович	2	3	4	4
27					
(Лічильник)		0	0	0	0

Record: 1 of 27

3. Створюємо таблицю, яка була б списком викладачів.

Побудуємо таблицю «Список_Викладачів» „з нуля". Нехай вона міститиме такі поля: **ID** типу **AutoNumber**, **ПІП викладача** типу **Text**, **Навчальна дисципліна** типу **Text**. Знаходячись у контейнері БД за умови, що активним пунктом у меню **Objects** є пункт **Tables**,

Список_Викладачів : Table

ID	ПІП викладача	Навчальна дисципліна
1	Домбровський І.В.	Вища математика
2	Мушак А.Я.	Інформатика
3	Шумка М.Л.	Філософія
4	Яценко І.І.	Історія економічних вчень
(Aut		

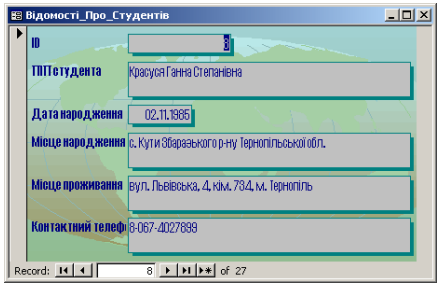
Record: 5 of 5

цокаємо на кнопці **New** панелі інструментів, розташованої вздовж верхнього краю вікна та у діалоговому вікні **New Table** вибираємо пункт **Design View**. З'являється вікно, у якому формуватимемо структуру таблиці «Список_Викладачів». Зазначаємо назви полів та їх типи.

Далі переходимо з вікна структури у режим таблиці. При цьому програма попросить зберегти таблицю, для якої лише структура є готовою. Заповнюємо комірки таблиці.

4. *Створюємо форми для перегляду та редагування даних таблиць.*

Покажемо, як побудувати форму для перегляду та редагування даних таблиці «Відомості_Про_Студентів». У меню **Objects** контейнера БД вибираємо пункт **Forms**. Далі цокаємо на кнопці **New** панелі інструментів, розташованої вздовж верхнього краю вікна контейнера та у діалоговому вікні **New Form** вибираємо спосіб створення форми, наприклад, **AutoForm: Columnar**, а також зазначаємо таблицю «Відомості_Про_Студентів», елементи якої „працюватимуть" у формі.



За допомогою системи керування записами форми переходимо на наступний запис; на тринадцятий запис; на останній запис таблиці. Збережемо побудовану форму, давши їй ім'я «Форма_ Відомості_Про_Студентів».

Побудова форм для перегляду та редагування даних решта таблиць є аналогічною.

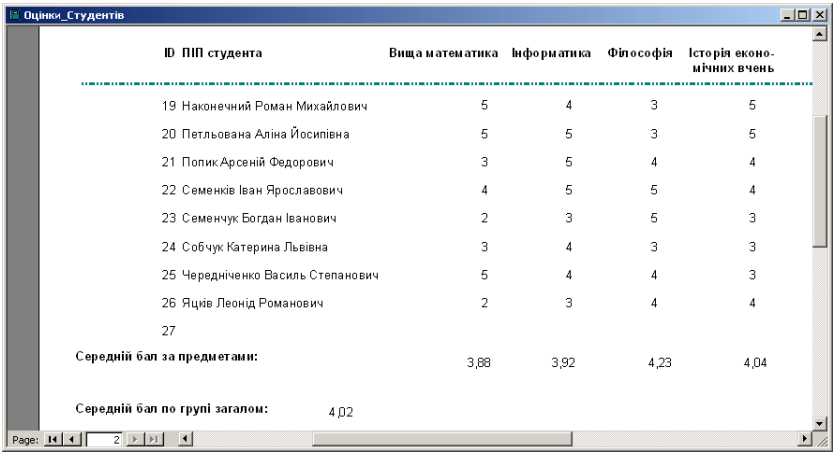
5. *Визначаємо середній бал по групі за кожним із предметів і загалом середній бал по групі.*

Для розв'язання цієї задачі побудуємо звіт, у якому працювали б формули, які обчислювали б середні значення. У меню **Objects** контейнера БД вибираємо пункт **Reports**. Далі цокаємо на кнопці **New** панелі інструментів, розташованої вздовж верхнього краю вікна контейнера та у діалоговому вікні **New Report** вибираємо спосіб створення звіту, зокрема, **AutoReport: Tabular**, а також зазначаємо таблицю «Оцінки_Студентів», на основі якої будуватиметься звіт. Отриманий звіт потрібно модифікувати так, щоб він містив необхідні формули та інші записи. Для цього переходимо у режим конструктора звітів (**Design View**) та в область **Report Footer** додаємо такі елементи керування: **написи** „Середній бал за

предметами:” і „Середній бал по групі загалом:” та 5 **полів**, 4 з яких призначені для обчислення середнього балу з кожного предмету, а п’яте – для обчислення середнього балу по групі. Формули для обчислення середнього балу з кожного предмету відрізнятимуться між собою лише аргументом функції **Avg**, котрий є назвою поля таблиці «Оцінки_Студентів», як-от, [Вища математика], [Інформатика], [Філософія], [Історія економічних вчень]. Наприклад, формула для обчислення середнього балу з інформатики матиме вигляд: **=Round(Avg([Інформатика]);2)**. У цій формулі **Avg** – функція, яка обчислює середнє значення зі всіх значень поля Інформатика, **Round** – функція, яка округлює знайдене середнє значення до двох знаків після коми, на що вказує другий аргумент. Формула, яка „працює” в останньому **полі** така:

=Round((Round(Avg([Вища математика]);2)+Round(Avg([Інформатика]);2)+Round(Avg([Філософія]);2)+Round(Avg([Історія економічних вчень]);2))/4;2)

Формули записують у розділі **Control Source** вікна властивостей встановлених елементів керування типу **поле**.

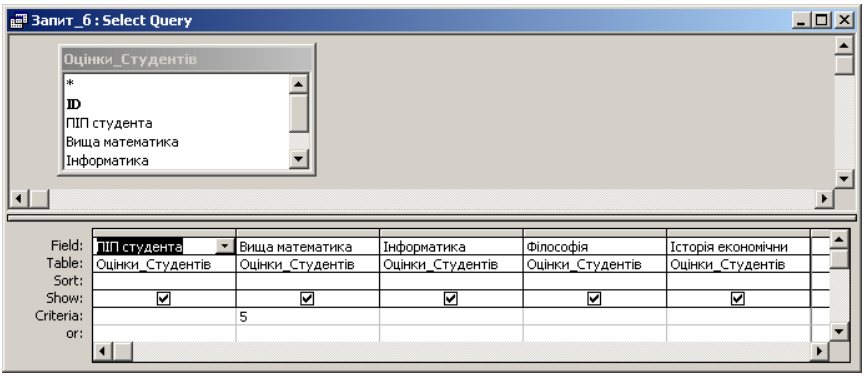


6. *Визначаємо середній бал по групі серед тих, хто має найвищу оцінку з вищої математики*

Дана підзадача схожа на попередню з тією відмінністю, що в ролі первинних даних для побудови звіту виступатиме не вся таблиця «Оцінки_Студентів», а лише її „екстракт” – записи про студентів, які отримали з вищої математики п’ятірку. Отже, перш ніж почати будувати

звіт із необхідними формулами, необхідно побудувати запит на вибірку, який би відокремив інформацію про потрібних студентів.

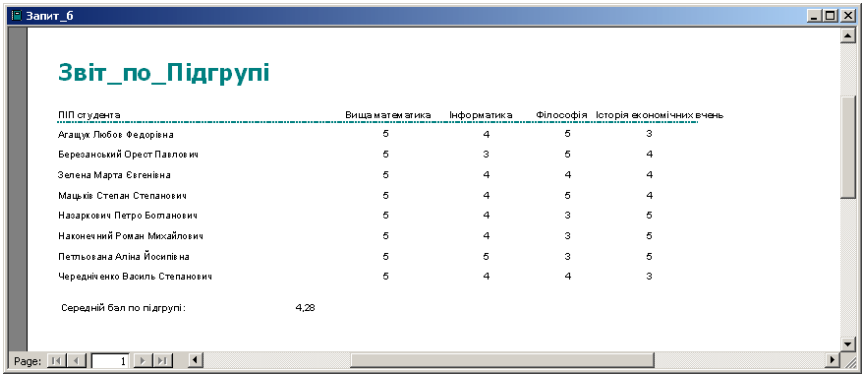
Знаходячись у контейнері БД за умови, що активним пунктом у меню **Objects** є пункт **Queries**, цокаємо на записі **Create query in Design View** панелі, котра відображає список об'єктів БД. Далі з'являються два вікна. В активному вікні (**Show Table**) вибираємо таблицю «Оцінки_Студентів», на основі полів якої будуватимемо потрібний запит. Закриваємо дане вікно. Для шуканого запиту встановлюємо потрібні поля. Для поля Вища математика записуємо критерій **=5**.



Запускаємо спроектований запит на виконання командою **Query→Run**. Зберігаємо запит.

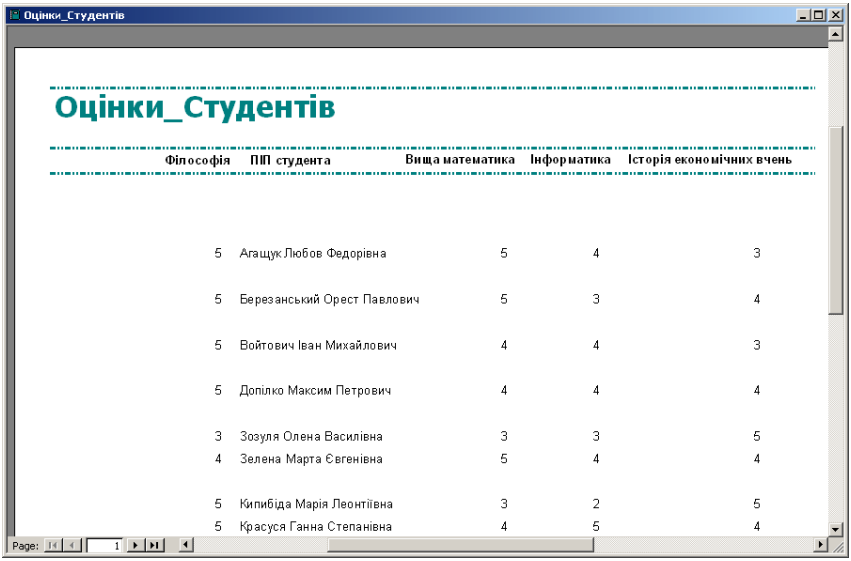
Виконуємо побудову звіту на основі щойно побудованого запиту, взявши за основу технологію, описану у підзадачі 5. Середній бал за предметами немає потреби обчислювати; нас цікавить лише середній бал по підгрупі загалом. Формула для обчислення шуканого значення така:

=Round((Round(Avg([Вища математика]);2)+Round(Avg([Інформатика]);2)+Round(Avg([Філософія]);2)+ Round(Avg([Історія економічних вчень]);2))/4;2)



7. *Створюємо звіт по всіх студентах; групування проведемо за першою літерою прізвища студента; сортування всередині групи проведемо за оцінкою з філософії*

Знаходячись у контейнері БД за умови, що активним пунктом у меню **Objects** є пункт **Reports**, виконуємо подвійне цокання на записі **Create report by using wizard** панелі, котра відображає список об'єктів БД. Далі з'являється перше вікно майстра **Report Wizard**. У ньому вибираємо таблицю «Оцінки_Студентів», на основі полів якої будуватимемо потрібний звіт. Вибираємо потрібні поля – усі, крім поля **ID**. Переходимо до наступного вікна майстра. Вибираємо поле для групування; згідно умови задачі це поле – **ПІП студента**. Далі цокаємо на кнопці **Grouping Options …** для встановлення характеру використання вибраного поля. З'являється діалогове вікно **Grouping Intervals**. Зі списку, який розкривається, вибираємо значення **1st Letter** і переходимо до третього вікна майстра звітів. Тут вибираємо порядок сортування. Сортуватимемо дані в групі за полем **Філософія** в абетковому порядку. Опції, які знаходяться у двох наступних вікнах майстра дозволяють змінити лише зовнішній вигляд звіту. Вказані за замовчуванням значення цих опцій нас задовільняють. В останньому вікні майстра задаємо назву звіту.



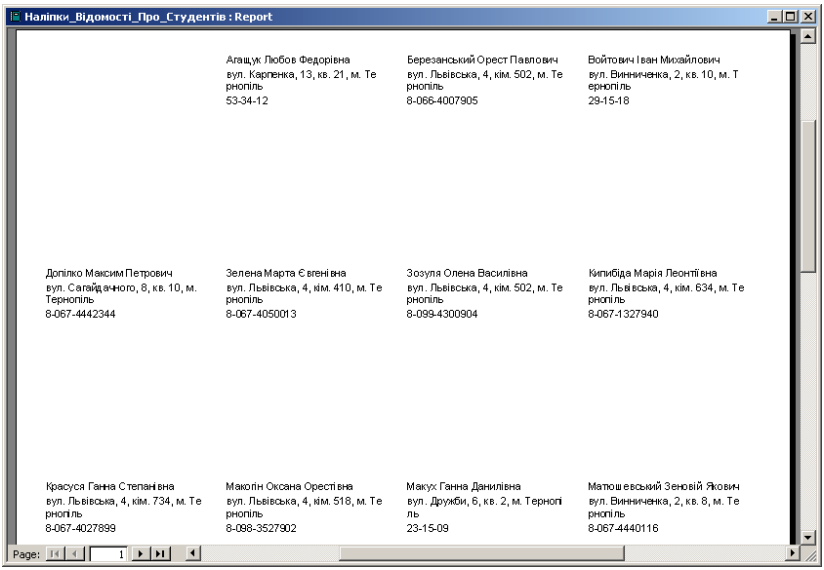
При потребі переходимо в режим конструктора (**Design View**) та підкоректовуємо розташування елементів керування у звіті.

8. Створюємо поштові наліпки для усіх студентів групи

Наліпки в Access створюють за допомогою звітів. Найкращий спосіб створення наліпок – використати майстер наліпок.

Знаходячись у контейнері БД за умови, що активним пунктом у меню **Objects** є пункт **Reports**, цокаємо на кнопці **New** панелі інструментів, розташованої вздовж верхнього краю вікна. З'являється діалогове вікно **New Report**. Вибираємо пункт **Label Wizard** і вказуємо таблицю «Відомості_Про_Студентів», на основі якої будуть підготовлені наліпки. Перше вікно майстра дозволяє, зокрема, вибрати розмір наліпки з усіх заготовок виробника наліпок. Нехай виробником є фірма **RankXerox**, а розмір наліпки такий: **89x47 мм**. Наступне вікно майстра дозволяє, зокрема, вибрати шрифт, його розмір і колір тексту. Значення цих параметрів залишаємо заданими за замовчуванням. У третьому вікні майстра вказують поля таблиці, які будуть розміщені на поштовій наліпці. Можна також додати пропуски, довільний текст, порожні рядки та знаки пунктуації. Вибрати поля можна або за допомогою подвійного цокання мишкою, або виділивши поле у списку **Available Fields** і цокнувши на кнопці >, розміщеній між двома списками. Нехай наліпки містять такі поля: **ПІП студента**, **Місце проживання** і **Контактний телефон** і кожне із них

починається з нового рядка. Наступне вікно майстра дозволяє відсортувати наліпки за одним чи декількома полями таблиці. Відсортуємо наліпки за полем **ПІП студента**. Виділяємо це поле у списку **Available Fields** і цокаємо на кнопці >. Останнє вікно майстра пропонує задати ім'я звіту – нехай **Наліпки_Відомості_Про_Студентів**. Встановимо також опцію **See the labels as they will look printed**.



9. *Готуємо привітальні листи всім студентам, роздрукувавши їх оцінки, середній бал та рейтинг (середній бал відносно середнього по групі, помножений на 100)*

В цілому зміст привітальних листів є сталим. „Змінними величинами", які характеризують того чи іншого студента в цих листах є такі елементи: 1. **Прізвище, ім'я, по батькові студента**, 2. **Оцінка з вищої математики**, 3. **Оцінка з інформатики**, 4. **Оцінка з філософії**, 5. **Оцінка з історії економічних вчень** та ще два поля, які не є складовими таблиці «Оцінки_Студентів»: **Середній бал** і **Рейтинг**. Отже, перш ніж послуговуватись майстром злиття з документами Microsoft Word, потрібно виконати таку підготовчу роботу – створити запит на основі таблиці «Оцінки_Студентів», який би містив усі поля даної таблиці та ще два обчислювальні поля – **Середній бал** і **Рейтинг** та на завершення підготувати у зазначеному вище текстовому редакторі заготовку листа.

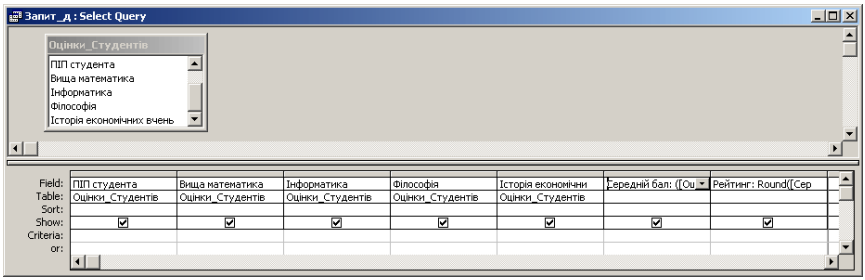
Знаходячись у контейнері БД за умови, що активним пунктом у меню **Objects** є пункт **Queries**, цокаємо на записі **Create query in Design View** панелі, котра відображає список об'єктів БД. Далі з'являються два вікна. В активному вікні (**Show Table**) вибираємо таблицю «Оцінки_Студентів», на основі полів якої будуватимемо потрібний запит. Закриваємо дане вікно. Для шуканого запиту встановлюємо потрібні поля. Зокрема, вираз для знаходження значень обчислювального поля **Середній бал** такий:

Середній бал: ([Оцінки_Студентів]![Вища математика]+ [Оцінки_Студентів]![Інформатика]+[Оцінки_Студентів]![Філософія]+[Оцінки _Студентів]+[Історія економічних вчень])/4

а вираз для знаходження значень обчислювального поля **Рейтинг** такий:

Рейтинг: Round([Середній бал]/4,02*100;2)

В останньому виразі константа 4,02 – це значення середнього балу по групі загалом, отримане в ході виконання підзадачі 5.



Запускаємо спроектований запит на виконання командою **Query→Run**. Зберігаємо запит.

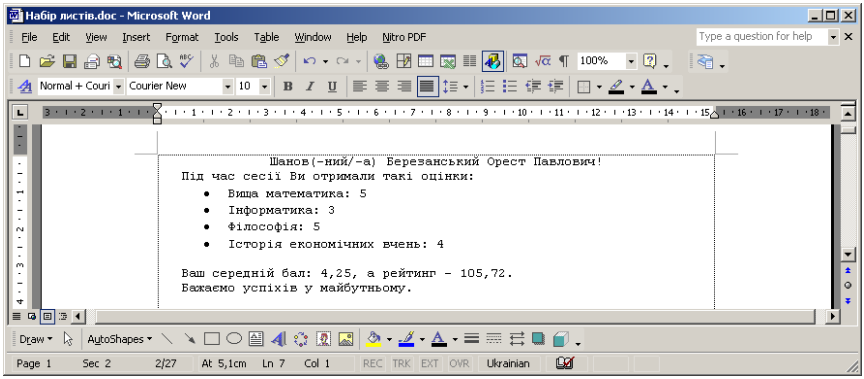
За допомогою текстового редактора Microsoft Word готуємо документ (заготовку листа), у якому символами ххх позначаємо ті місця, які є унікальними для кожного студента.

Шанов(-ний/-а) ххх!
Під час сесії Ви отримали такі оцінки:

- Вища математика: ххх
- Інформатика: ххх
- Філософія: ххх
- Історія економічних вчень: ххх

Ваш середній бал: ххх, а рейтинг – ххх.
Бажаємо успіхів у майбутньому.

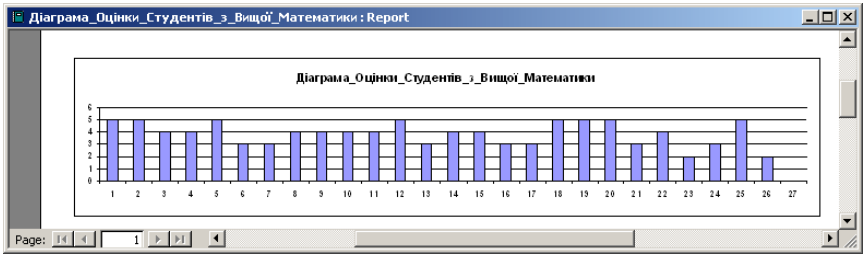
Підготовча робота закінчена. Далі, знаходячись у контейнері БД за умови, що активним пунктом у меню **Objects** є пункт **Queries**, вибираємо підготовлений запит і цокаємо на направленій униз стрілці, розміщеній поруч із кнопкою **OfficeLinks** панелі інструментів **Database**. У меню, яке з'явилося вибираємо команду **Merge It with Microsoft Word**. З'являється діалогове вікно майстра. Вибираємо опцію **Link your data to an existing Microsoft Word document**. Access відобразить стандартне вікно вибору файлів у Windows. Вибираємо файл, який є заготовкою листа. В результаті запускається текстовий редактор Word та активізується режим злиття. Тепер модифікуємо отриманий документ. Послуговуючись областю задач **Mail Merge** текстового редактора переходимо до четвертого кроку. А далі виділяємо перше входження символів ххх у документ, натискуємо пункт **More items…** області задач та у діалоговому вікні **Insert Merge Field**, яке з'явилося, вибираємо назву потрібного поля. Аналогічні дії повторюємо для решти символів ххх. Переходимо до шостого кроку області задач. Виконуємо команду **Edit individual letters…** і у діалоговому вікні **Merge to New Document** вибираємо опцію **All**. Набір привітальних листів студентам готовий; кожен лист міститься на новій сторінці. Зберігаємо згенерований текстовим редактором файл.



10. Будуємо гістограму розподілу оцінок із вибраної користувачем дисципліни

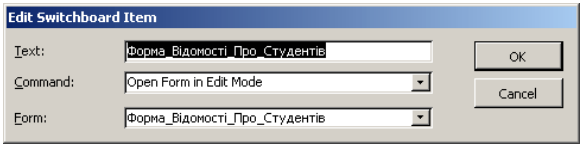
Гістограми по-іншому називають стовбчиковими діаграмами. Скористаємось майстром діаграм для побудови шуканої гістограми. Активізуємо контейнер БД. Нехай активним пунктом у меню **Objects** є пункт **Reports**. Цокаємо на кнопці **New** панелі інструментів, розташованої вздовж верхнього краю вікна. З'являється діалогове вікно **New Report**. Вибираємо пункт **Chart Wizard** і вказуємо таблицю «Оцінки_Студентів», на основі якої буде створена гістограма. У першому вікні майстра задають поля, котрі містять дані, необхідні для побудови діаграми. Вибираємо такі поля: **ID** та, наприклад, **Вища математика**. Для цього виконують подвійне цокання мишкою на зазначених полях або послуговуються кнопкою >, розміщеною між двома областями: **Available Fields** та **Fields for Chart**. Наступне вікно пропонує вибрати тип діаграми з переліку. Вибираємо **Column Chart** (цей тип є активний за замовчуванням). Наступне вікно пропонує переглянути ескіз майбутньої діаграми. Останнє вікно майстра пропонує задати ім'я діаграми – нехай «Діаграма_Оцінки_Студентів_з_Вищої_Математики»; задаємо опцію не відображати легенду.

При потребі задаємо альбомну орієнтацію аркуша, встановлюємо якомога менші розміри полів (кнопка **Setup** панелі інструментів **Print Preview**), а також змінюємо розмір діаграми, відкривши її в режимі конструктора (**Design View**).



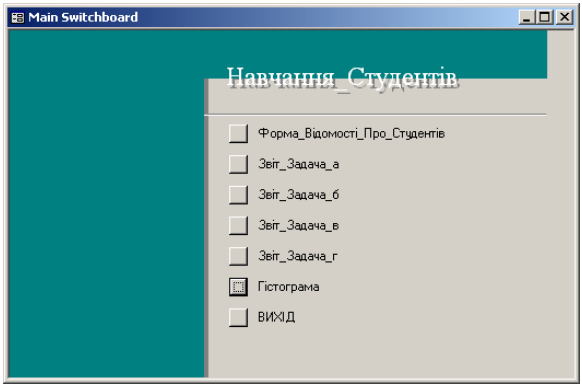
11. *Створюємо меню для зручної роботи користувача у вигляді форми, яка завантажуватиметься автоматично при відкритті БД*

Для побудови потрібної форми запускаємо диспетчер кнопкових форм – команда **Tools→Database Utilities→Switchboard Manager**. З'являється вікно, у якому даємо команду редагувати головну кнопкову форму (цокаємо на кнопці **Edit…**). В результаті відкривається вікно **Edit Switchboard Page**. Цокаємо на кнопці **New…** для встановлення першого пункту меню. В результаті переходимо до нового діалогового вікна **Edit Switchboard Item**. У цьому вікні у відповідних полях записуємо формулювання пункту меню, вибираємо команду із переліку команд, який організований у вигляді списку, що розкривається, та встановлюємо потрібні об'єкти БД. Останню кнопку організовуємо як вихід із форми.



На завершення задаємо опцію, яка дозволяє автоматично виводити на екран створену форму при відкритті БД. Для цього виконуємо команду **Tools→Startup…** і у розділі **Display Form/Page** із списку, який розкривається, вибираємо пункт **Switchboard**.

Шукана кнопкова форма готова.



Задача 2:

Є дані про клієнтів банку.

Оперативні дані	Довідкові дані
Прізвище, ім'я, по батькові клієнта Номер рахунку Сума внеску Термін збереження внеску (у місяцях) Дата внеску	Номер рахунку Банківський відсоток

Створити форми для введення і редагування даних кожної таблиці. Розрахувати суму, яка нараховується за банківським відсотком та загальною сумою до виплати по кожному клієнту; результат упорядкувати за термінами збереження внесків. Видати підсумкову відомість, яка містить мінімальну, максимальну та середню суму внеску по всіх клієнтах. Одержати дані за запитом із клавіатури:

- про клієнтів, термін збереження внесків яких становить 3, 6, 10 місяців;
- за конкретною датою внеску.

Створити кнопкову форму для роботи із завданням.

Розв'язання задачі 2:

1. Створюємо нову БД «Клієнти_Банку».

Після запуску MS Access з'являється область задач **New File**. У групі опцій **New** вибираємо пункт **Blank Database**. Далі Access пропонує зберегти БД. Вибираємо потрібну папку та вводимо назву БД «Клієнти_Банку».

2. Створюємо дві таблиці: «Оперативні_Дані» та «Довідкові_Дані».

Будуємо таблицю «Оперативні_Дані». Беручи до уваги умову задачі, ця таблиця міститиме такі поля: «Прізвище, ім'я, по батькові клієнта» типу Text, «Номер рахунку» типу Number, «Сума внеску» типу Number, «Термін збереження внеску» типу Number, «Дата внеску» типу Date/Time.

Оперативні_Дані : Table

Прізвище, ім'я, по батькові клієнта	Номер рахунку	Сума внеску	Термін збереження внеску	Дата внеску
Антощук Володимир Іванович	123456	13015	17 міс.	12.12.2006
Барило Семен Петрович	408909	22000	34 міс.	05.05.1999
Борисюк Василь Петрович	788020	140800	22 міс.	04.08.2005
Вавринець Діана Іванівна	100059	12000	18 міс.	12.09.2006
Гакало Іван Павлович	199086	23000	13 міс.	13.06.2005
Дубівець Ніна Степанівна	588401	99500	20 міс.	14.10.2006
Єрьоменко Інесса Марківна	100276	15000	9 міс.	12.09.2006
Йордан Людмила Євстахівна	212088	45000	13 міс.	11.02.2004
Колодій Марія Михайлівна	131066	23870	16 міс.	14.10.2006
Лешків Зоряна Василівна	100085	9000	36 міс.	16.09.2006
Маслів Зеновій Михайлович	522016	10700	10 міс.	22.03.2005
Мовчазний Віталій Борисович	444090	15000	7 міс.	23.10.2005
Навроцький Василь Богданович	309112	30225	11 міс.	16.08.2006
Тимчій Василина Іванівна	199004	12000	15 міс.	23.09.2004
Шемлей Катерина Борисівна	200016	3000	150 міс.	24.09.2004
	0	0	міс.	

Record: 1 of 15

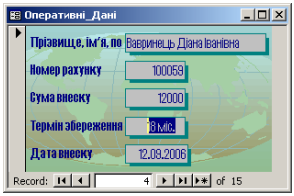
Знаходячись у контейнері БД за умови, що активним пунктом у меню **Objects** є пункт **Tables**, цокаємо на кнопці **New** панелі інструментів, розташованої вздовж верхнього краю вікна та у діалоговому вікні **New Table** вибираємо пункт **Design View**. З'являється вікно, у якому формуватимемо структуру таблиці. Зазначаємо назви полів та їх типи. Для поля «Термін збереження внеску» у розділі **Field Properties** зазначаємо формат – #" міс.".

Далі переходимо з вікна структури у режим таблиці (**Datasheet View**). При цьому програма попросить зберегти таблицю, для якої лише структура є готовою. Врешті-решт, заповнюємо комірки таблиці.

Побудову таблиці «Довідкові_Дані» виконують аналогічно до побудови таблиці «Оперативні_Дані».

3. *Створюємо форми для введення і редагування записів кожної таблиці.*

Знаходячись у контейнері БД за умови, що активним пунктом у меню **Objects** є пункт **Forms**, цокаємо на кнопці **New** панелі інструментів, розташованої вздовж верхнього краю вікна та у діалоговому вікні **New Form** вибираємо пункт **AutoForm: Columnar** і вказуємо таблицю «Оперативні_Дані», на основі якої будуватимемо форму. Після побудови форми зберігаємо її під іменем «Форма_Оперативні_Дані».



Форма для роботи із таблицею «Довідкові_Дані» будується аналогічно.

4. *Розраховуємо суму, яка нараховується за банківським відсотком та загальною сумою до виплати по кожному клієнту; результат упорядкуємо за термінами збереження внесків.*

Створимо запит з обчислювальним полем, у якому Access згенерує шукану суму для кожного клієнта банку. Нехай цей запит міститеме такі поля: «Прізвище, ім'я, по батькові клієнта», «Номер рахунку», «Термін збереження внеску» та обчислювальне поле, назва якого «Шукана сума». Найбільший інтерес при побудові даного запиту має вираз для знаходження значень обчислювального поля. Запишемо математичну модель для знаходження потрібної суми.

Нехай CB – сума внеску, $БВ$ – значення банківського відсотку, TB – термін вкладу, C – шукана сума. Тоді формула для знаходження шуканої суми матиме вигляд:

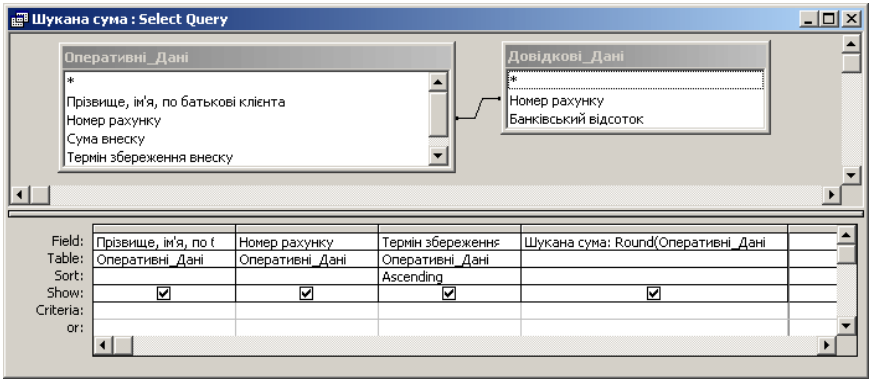
$$C = CB + CB \cdot БВ \cdot \frac{TB}{12}$$

„Перекладемо” цю математичну формулу мовою Access. Маємо:

Шукана сума: Round([Оперативні_Дані]![Сума внеску]+[Оперативні_Дані]![Сума внеску]*([Довідкові_Дані]![Банківський відсоток]/100)*[Оперативні_Дані]![Термін збереження внеску]/12;2)

Тут функція Round заокруглює значення шуканої суми до двох знаків після коми.

Знаходячись у контейнері БД за умови, що активним пунктом у меню **Objects** є пункт **Queries**, цокаємо на записі **Create query in Design View** панелі, котра відображає список об’єктів БД. Далі з’являються два вікна. В активному вікні (**Show Table**) вибираємо дві таблиці «Оперативні_Дані» та «Довідкові_Дані», на основі полів яких будуватимемо потрібний запит. Закриваємо дане вікно. Створюємо зв’язок між цими таблицями, кожна з яких має поле «Номер рахунку». Для шуканого запиту встановлюємо потрібні поля.



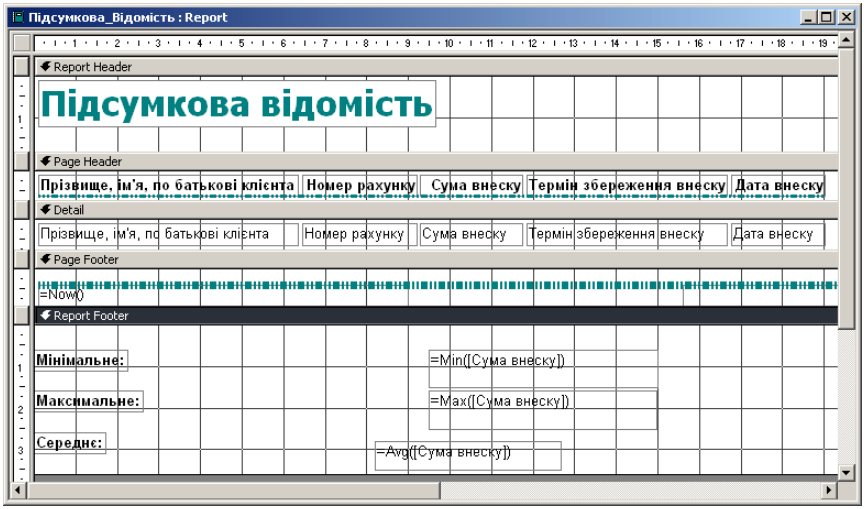
На завершення задаємо сортування отриманих записів за полем «Термін збереження внеску» у порядку зростання. Запускаємо спроектований запит на виконання командою **Query→Run**. Зберігаємо запит під іменем «Шукана сума».

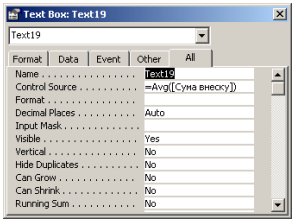
5. *Готуємо підсумкову відомість, яка містить мінімальну, максимальну та середню суму внеску по всіх клієнтах.*

У термінах СУБД Access така відомість має ім’я – звіт. Побудуємо його. Знаходячись у контейнері БД за умови, що активним пунктом у меню **Objects** є пункт **Reports**, цокаємо на кнопці **New** панелі інструментів, розташованої вздовж верхнього краю вікна. З’являється діалогове вікно **New Report**. Вибираємо пункт **AutoReport: Tabular** і вказуємо таблицю

(«Оперативні_Дані»), на основі якої будуватимемо звіт. Зберігаємо його під іменем «Підсумкова_Відомість».

Звіт готовий, але він не містить значень мінімальної, максимальної та середньої суми внеску по всіх клієнтах. Модифікуємо створений звіт. Для цього переходимо у режим **Design View** і в області **Report Footer** будуємо три пари елементів управління **Label** та **Text Box** (по парі для мінімального, максимального та середнього значення сум внесків). Для цього попередньо цокнувши мишкою на піктограмі **Text Box** у вікні **Toolbox**, рисуємо кожну пару в області **Report Footer**. Нехай кожен елемент управління типу **Label** міститься відразу біля лівого краю звіту, а кожен **Text Box** – під значеннями поля «Сума внеску».





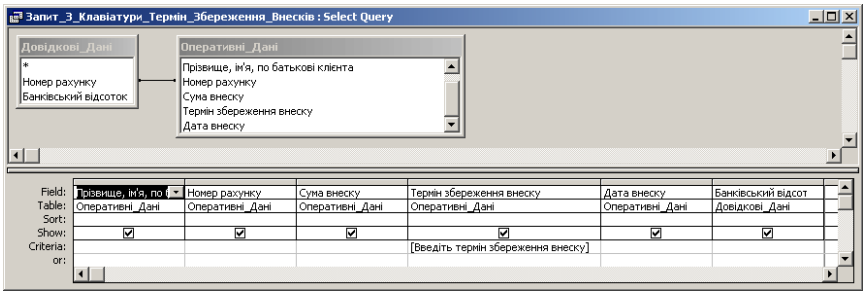
Для знаходження шуканих значень система послуговується формулами, які записують у розділі **Control Source** вікна властивостей встановлених елементів керування типу **Text Box**. Зокрема, для мінімального значення маємо формулу ***=Min([Сума внеску])***, для максимального значення ***=Max([Сума внеску])***, а для середнього значення ***=Avg([Сума внеску])***. Переходимо в режим відображення звіту.

6. *Одержуємо дані за запитом із клавіатури:*

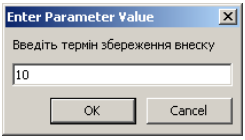
- *про клієнтів, термін збереження внесків яких становить 3, 6, 10 місяців;*
- *за конкретною датою внеску.*

Запит із клавіатури по-іншому називають запитом з параметрами. Побудуємо такого типу запит, параметром якого є термін збереження внесків.

Переходимо у режим конструктора запитів (**Design View**). Запит формуватимемо на основі таблиць «Оперативні_Дані» та «Довідкові_Дані». Нехай він міститиме усі поля першої таблиці та поле «Банківський відсоток» таблиці «Довідкові_Дані». У розділі **Criteria** поля «Термін збереження внеску» формуємо параметр =[Введіть термін збереження внеску].



Запускаємо спроектований запит на виконання командою **Query→Run**. З'являється діалогове вікно **Enter Parameter Value**, у якому вказуємо значення терміну збереження внеску, наприклад 10. В результаті система виведе інформацію лише про тих клієнтів банку, термін збереження внеску яких дорівнює 10 місяців.



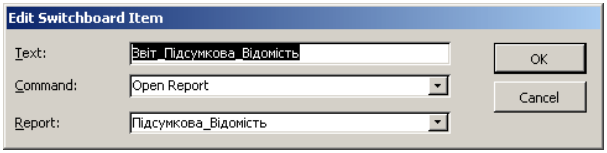
Друга частина задачі є цілком аналогічною до першої. Нехай запит, який є результатом її розв'язання містить ті ж самі поля, що і попередній запит, а значення поля «Дата внеску» встановлюються на основі введених з клавіатури даних. У режимі конструктора запитів (**Design View**) формуємо параметр =[Введіть дату внеску]. Запускаємо запит на виконання і у вікні, що з'явилося, вказуємо конкретну дату внеску.

7. Створюємо кнопкову форму для роботи із задачею.

Призначенням даної кнопкової форми є реалізація меню для виклику двох побудованих раніше форм і звіту. Останній пункт меню кнопкової форми повинен давати змогу закрити її.

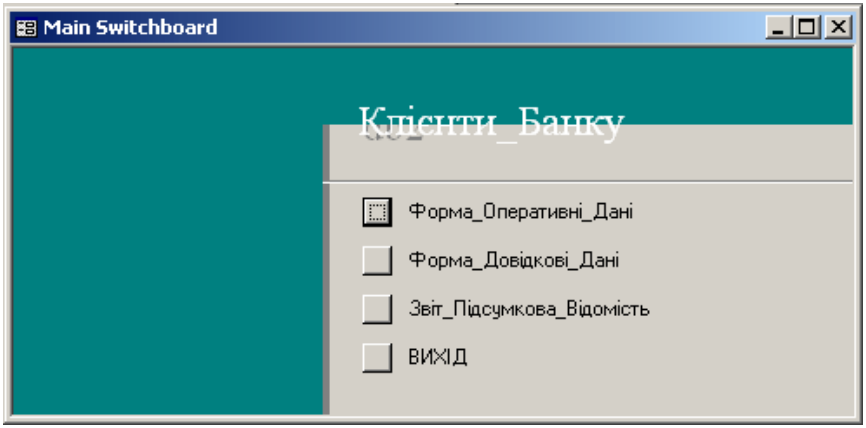
Запускаємо диспетчер кнопкових форм – команда **Tools→Database Utilities→Switchboard Manager**. З'являється вікно, у якому даємо команду редагувати головну кнопкову форму (цокаємо на кнопці **Edit…**). В результаті відкривається вікно **Edit Switchboard Page**. Цокаємо на кнопці

New… для встановлення першого пункту меню. В результаті переходимо до нового діалогового вікна **Edit Switchboard Item**. У цьому вікні у відповідних полях записуємо формулювання пункту меню, вибираємо команду із переліку команд, який організований у вигляді списку, що розкривається, та встановлюємо потрібні об'єкти БД. Останню кнопку організовуємо як вихід із форми.



На завершення задаємо опцію, яка дозволяє автоматично виводити на екран створену форму при відкритті БД. Для цього виконуємо команду **Tools→Startup…** і у розділі **Display Form/Page** із списку, який розкривається, вибираємо пункт **Switchboard**.

Шукана кнопкова форма готова.



Задача 3:

Розміри вкладів і кредитів групи юридичних і фізичних осіб наведені у БД „Банківські капітали”:

	B	C	D	E	F	G
5	№ з/п	Прізвище І.П. клієнта або назва установи	Величина кредиту	Термін погашення	Розмір депозиту	Термін вкладання
6	1	ВТО „Ватра”	240,000.00 грн.	20/02/2003	460,000.00 грн.	27/01/1995
7	2	ВО „Текстерно”	124,534.00 грн.	25/11/2004	342,321.00 грн.	08/06/1997
8	3	Петренко І.С.	1,200.00 грн.	11/11/2002	2,400.00 грн.	06/10/1994
9	4	Завод „Оріон”	45,905.00 грн.	22/04/2001	76,543.00 грн.	29/07/1997
	…	…	…	…	…	
19	14	Семків І.І.	2,300.00 грн.	22/12/2003	-	
20	15	ТОВ „Неотек”	2,960.00 грн.	27/07/2005	23,090.00 грн.	24/11/1996
21	16	АТ „Технотерн”	34,700.00 грн.	09/01/2006	340,560.00 грн.	24/06/1993

Виконати такі завдання:

1) Розрахувати середню величину депозиту клієнтів банку. Знайти вклади, котрі перевищують середній і вивести їх окремою таблицею.
2) Розрахувати середню величину кредиту клієнтів банку, виданих з терміном погашення до 2003 року включно. Знайти кількість кредитів, які менші розрахованого значення і вивести їх окремою таблицею.
3) Розрахувати суму кредитів, виданих до 2001 року включно. Знайти актив банку, як різницю між сумою депозитів і кредитів.

Розв’язання задачі 3:

1) Для розрахунку середньої величини депозитів клієнтів банку заносимо у клітинки B23 та E23 такі записи:

Середня величина депозиту, =AVERAGE(F6:F21)

відповідно.

Для знаходження вкладів, розмір яких перевищує середнє значення депозиту, виділяємо таблицю і виконуємо команди **Data (Данные)→Filter (Фильтр) →Autofilter(Автофильтр)**. Далі, використовуючи маркер автофільтру у колонці „Розмір депозиту”, вибираємо команду **Custom… (Условие…)** і у діалоговому вікні **Custom Autofilter (Пользовательский автофильтр)** задаємо критерій >=120,984.43. Отриману вибірку копіюємо в область B26:C30.

	B	C
26	№ п/п	Прізвище І.П. клієнта або назва установи
27	1	ВТО “Ватра”
28	2	ВО “Текстерно”
29	8	АСТ “Терен”
30	16	АТ “Технотерн”

2) Розрахунок середньої величини кредиту клієнтів банку, виданих з терміном погашення до 2003 року включно, проводимо за допомогою матричної формули {=AVERAGE(IF(E6:E21<=E32,D6:D21,""))} занесеної у клітинку C33, при цьому в клітинці E32 міститься дата 31/12/2003. Для знаходження кількості кредитів, які менші розрахованого значення 45640.29 грн., в клітинку C35 записуємо матричну формулу

{=COUNT(IF(E6:E21<=E32,IF(D6:D21<C33,1,"")),"")}

Таблицю кредитів, менших 45640.29 грн., будуємо аналогічно (див. підзадачу 1)).

3) Для обчислення суми кредитів, виданих до 2001 року включно, в клітинки B36 та E36 заносимо:

Сума кредитів, виданих до 2001 року включно,

=SUMIF(E6:E21,"<=12/31/01",D6:D21)

відповідно.

Щоб знайти актив банку, шукаємо різницю між сумами депозитів і кредитів у клітинці D38: =SUM(F6:F21)-SUM(D6:D21)

Задача 4:

Використовуючи БД „100 найбільших банків", виконати такі завдання:

1) створити список банків, у яких статутний фонд (Total Assets) знаходиться в діапазоні від 133% до 355% від середнього значення статутного фонду;
2) визначити сумарний власний капітал (Ownership) вибраних банків і порівняти його з власним капіталом п'ятого за величиною банку з точністю до трьох десяткових знаків;
3) для банків із утвореного списку визначити відношення прибутку (Benefit) до середнього значення прибутку з точністю до чотирьох десяткових знаків;
4) побудувати кругову діаграму для розподілу статутного фонду вибраних банків.

Розв'язання задачі 4:

1) Перш за все знаходимо межі вибірки, використовуючи БД „100 найбільших банків", розміщену в діапазоні (A4:E104):

	A	B	C	D	E
4	**Rank**	**Institution**	**Total Assets**	**Ownership**	**Benefit**
5	1	Cetecorp	738.2	600.34	73.82
6	2	Bank of Amereca	656.1	511.23	65.61
	…	…	…	…	…
103	99	Washengton Federal enc.	6.3	5.04	0.63
104	100	Valley Nateonal Corp.	6.2	4.96	0.62

Для цього заносимо дані та формули у такому вигляді:

	B	C	D
106	Середнє значення статутного фонду		=AVERAGE(C5:C104)
107	Нижня межа вибірки		=133%*D106
108	Верхня межа вибірки		=355%*D106

Шукані банки вибираємо за допомогою засобу **AutoFilter**.

	A	B	C	D	E
110	**Rank**	**Institution**	**Total Assets**	**Ownership**	**Benefit**
111	8	Washengton Mutual	188.6	150.88	18.86
112	9	FleetBoston F.C.	187.8	150.24	18.78
…	…	…	…	…	…
118	15	Bank of New York	76	60.8	7.6
119	16	PNC Bank	74.3	59.44	7.43
120	17	Ferstar	73	58.4	7.3

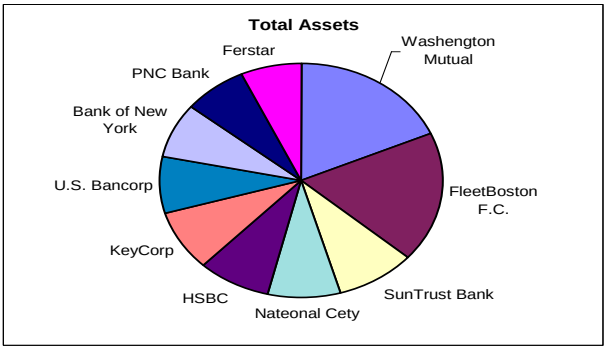
2) Визначаємо сумарний власний капітал (Ownership) вибраних банків, заносячи у клітинки A122 та C122 такі текст і формулу:

Сумарний власний капітал, =SUM(D111:D120)

Знаходимо відношення сумарного власного капіталу вибраних банків до власного капіталу п’ятого за величиною банку HSBS у клітинці C123: =C122/D115 *(результат = 11.984)*

3) Для визначення відношення прибутку (Benefit) до середнього значення прибутку заносимо у клітинки A124 та C124 такий текст і формулу: Середнє значення прибутку, =AVERAGE(E111:E120), а в

клітинку F111 – формулу =E111/C$124. Цю формулу копіюємо на область F112:F120.



4) За допомогою засобу „майстер діаграм” будуємо кругову діаграму для розподілу статутного фонду.

Задача 5:

Створити базу даних для обліку цінних паперів (акцій).

Розв’язання задачі 5:

Цінні папери – документи, які засвідчують право володіння і довготермінові зобов’язання емітентів щодо виплати їхнім власникам доходів; грошові й товарні документи, що засвідчують майнові права. До цінних паперів належать акції, облігації, векселі, чеки, депозити та інвестиційні сертифікати, акцепти, паї, ноти та інше.

Акція – вид цінного папера без встановленого терміну обігу, який свідчить про пайову участь у статутному фонді акціонерного товариства. Розрізняють такі види акцій: іменні, на пред’явника, привілейовані та прості.

Частина прибутку товариства, що залишилася після сплати дивідендів власником привілейованих акцій, розподіляється у вигляді дивідендів між власниками звичайних акцій.

Акціонери – це власники акцій акціонерних товариств. Акціонером може бути фізична або юридична особа, яка має право власності на акції.

Емітент має право на випуск акцій з моменту реєстрації цього випуску у відповідному фінансовому органі. Грошова сума, яка вказана в акції, називається номінальною вартістю акції, а ціна, згідно з якою акція реалізується на ринку – курс акції.

Використовується система реєстру власників іменних акцій – це сукупність даних, зафіксованих у паперовій та/або безпаперовій формі (у вигляді записів електронних баз даних), що забезпечує ідентифікацію зареєстрованих у цій системі власників, номінальних утримувачів іменних акцій, зареєстрованих на їхнє ім'я, одержання та надання інформації цим особам і складання реєстру власників іменних акцій.

Сертифікат акції – документ, який засвідчує право власності на придбані акції акціонерного товариства. Видається емітентом на сумарну номінальну вартість акцій, придбаних акціонером.

Вхідними документами для створення бази даних є заява, яку подає покупець на придбання акцій, та договір, який укладається між емітентом і покупцем.

Заява містить такі реквізити: код підприємства, назва підприємства, адреса підприємства, прізвище, ім'я та по батькові покупця, назва документа, який засвідчує особу, його серія та номер, адреса покупця.

В договорі вказуються такі реквізити: назва організації, прізвище, ім'я та по батькові та відповідні дані з документу, який засвідчує особу, кількість акцій, яку зобов'язується продати емітент, ціна однієї акції. загальна вартість акцій, адреса емітента та покупця.

Вихідними документами є: акція, сертифікат акцій, реєстр власників іменних акцій.

Для виділення об'єктів предметної області, які будуть зберігатися в базі даних, визначення характеристик інформаційних об'єктів, атрибутів, встановлення відношень між ними, які дозволяють реалізувати множину запитів до бази даних, будуємо агрегацію атрибутів в об'єкти.

Агрегування – об'єднання конструктивно незалежних елементів, які виконують різні функції, в єдину систему.

Перелік атрибутів подано в таблиці 1.

Таблиця 1

Перелік атрибутів

№ з/п	Атрибут	Опис
1.	Код власника	В реєстрі
2.	Назва власника	юридична або фізична особа
3.	Адреса власника	
4.	Документ	Паспорт
5.	№ і серія документа	
6.	Загальна вартість акцій	в Дороворі вказується сума, на яку продано акції
7.	Код емітента	в реєстрі
8.	Назва емітента	

9.	Адреса емітента	
10.	№ особового рахунку	
11.	Статутний фонд	
12.	Код реєстратора	в реєстрі
13.	Назва реєстратора	
14.	Адреса реєстратора	
15.	№ дозволу	
16.	Код країни	
17.	Назва країни	
18.	Дата закінчення дозволу	дозвіл видається на три роки
19.	№ емісії	
20.	Вид акцій	іменна, привілейована, на пред'явника
21.	Дата операції	
22.	Номінальна вартість	грошова сума, вказана в акції
23.	Кількість акцій	в акції та сертифікаті вказується кількість акцій, які випускаються акціонерним товариством
24.	Звітний період	
25.	Ставка дивіденду	в сертифікаті
26.	Вартість акцій	

Агрегація атрибутів в об'єкти

Перший крок агрегації.

Т(код власника, назва власника) =1:1
Т(код власника, адреса власника) =1:1
Отримано об'єкт "Власник"
Т(код емітента, адреса емітента) =1:1
Т(код емітента, назва емітента) =1:1
Отримано об'єкт "Емітент"
Т(код реєстратора, назва реєстратора) =1:1

Т(код реєстратора, адреса реєстратора) =1:1
Т(код реєстратора, № дозволу) =1:1
Т(код реєстратора, дата закінчення дозволу) =1:1
Отримано об'єкт "Реєстратор"

Другий крок агрегації.

Т(документ, Власник) =1:1
Т(№ і серія документа, Власник) =1:1
Т(код країни, Власник) =1:Б
Т(назва країни, Власник) =1:Б
Т(загальна вартість акцій, Власник) =1:Б
Ці атрибути включаємо в об'єкт "Власник"
Т(особовий рахунок, Емітент) =1:1
Т(статутний фонд, Емітент) =1:Б
Ці атрибути включаємо в об'єкт "Емітент"

Т(№ емісії, вид акцій) =1:Б

Т(код емітента, № емісії) =1:Б
Т(дата випуску, № емісії) =1:Б
Т(номінальна вартість, № емісії) =1:Б
Т(кількість акцій, № емісії) =1:Б

Отримано об’єкт “Емісія”

Т(код реєстратора, № сертифікату) =1:Б

Т(кількість акцій, № сертифікату) =1:Б
Т(вартість акцій, № сертифікату) =1:Б
Т(№ емісії, № сертифікату) =1:Б
Т(код власника, № сертифікату) =1:Б
Т(дата операції, № сертифікату) =1:Б

Отримано об’єкт “Сертифікат”

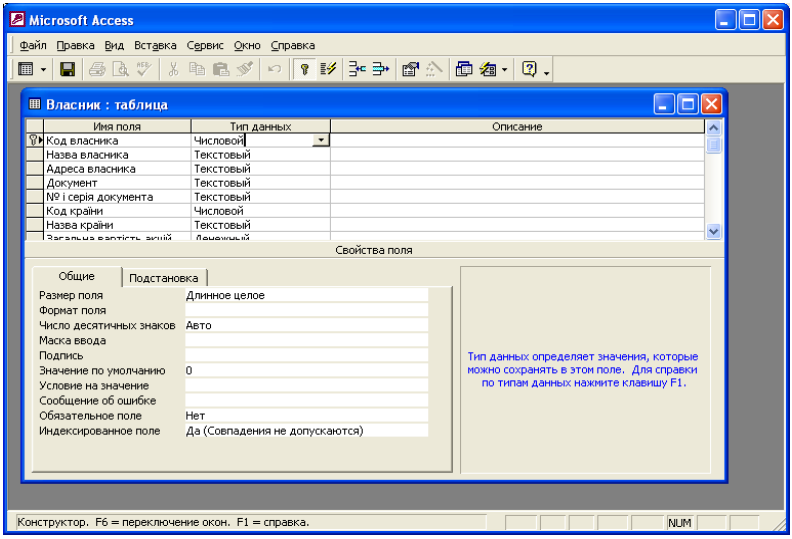
Т(код емітента, ставка дивіденту) =1:Б
Т(звітний період, № емісії) =1:Б

Отримано об’єкт “Дивіденди”

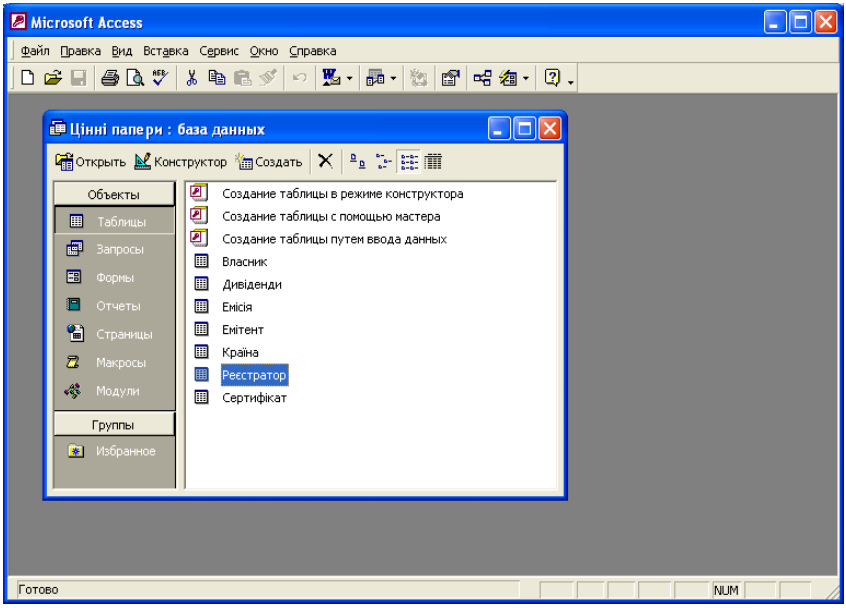
В результаті агрегації виділено такі об’єкти: Власник, Реєстратор, Емітент, Дивіденди, Емісія, Сертифікат.

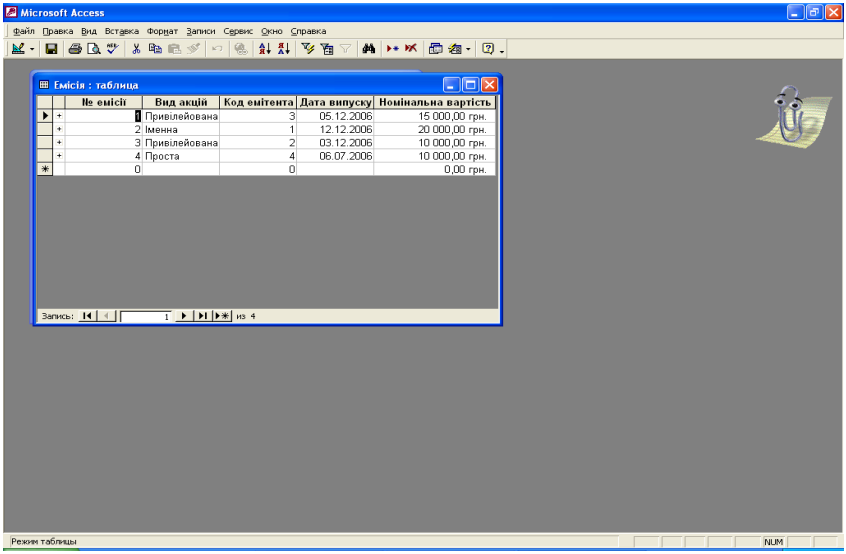
Для реалізації задачі вибрано систему керування базами даних MS Access, яка підтримує реляційну модель даних. Вона орієнтована на роботу з таблицями баз даних, формами, запитами, звітами, сторінками, макросами, модулями.

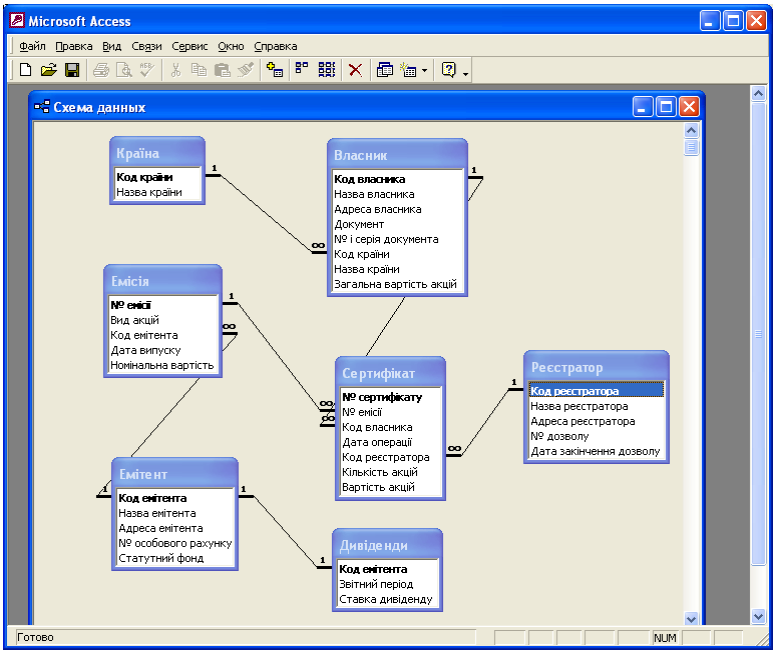
Таблиці баз даних створюються для збереження даних, які стосуються об’єктів предметної сфери. Об’єкти, утворені врезультаті агрегації, уводяться в СУБД Access як таблиці в режимі **Конструктора:**



Результат побудови структури усіх таблиць задачі, уведення даних та побудови схеми даних має вигляд:





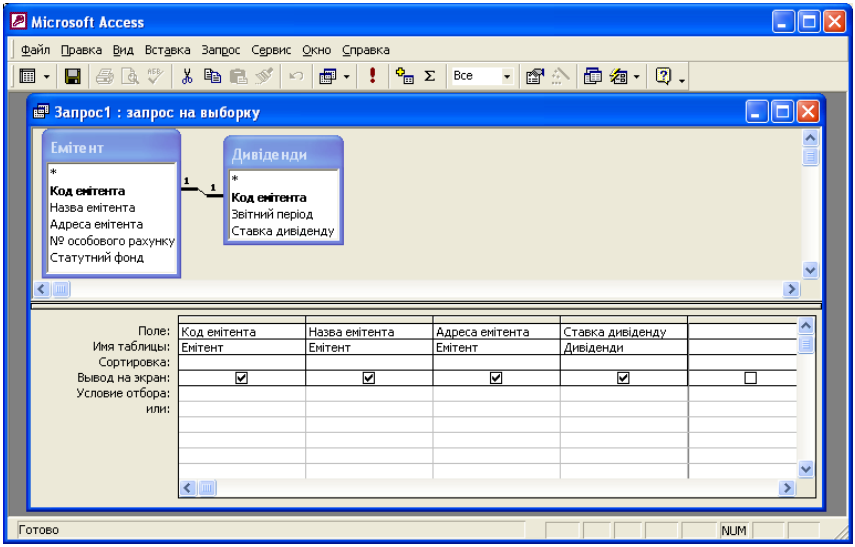


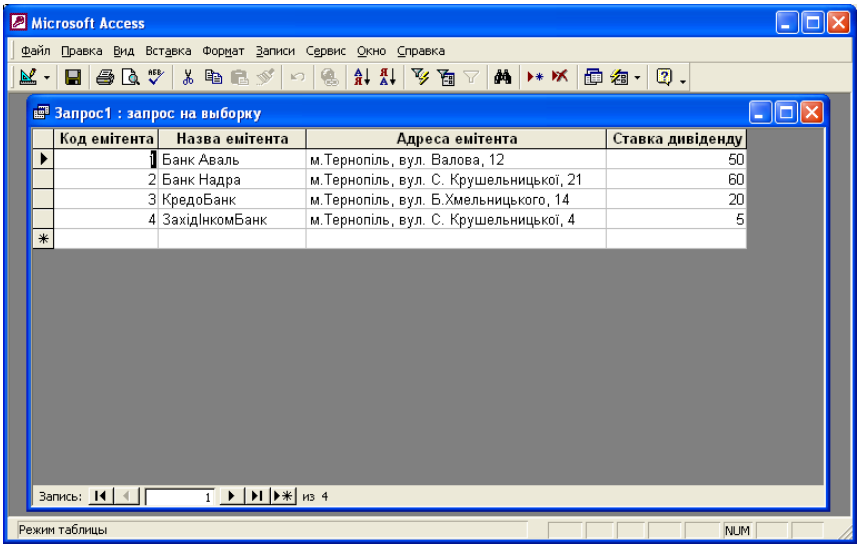
Таблиця 2

Перелік запитних зв’язків

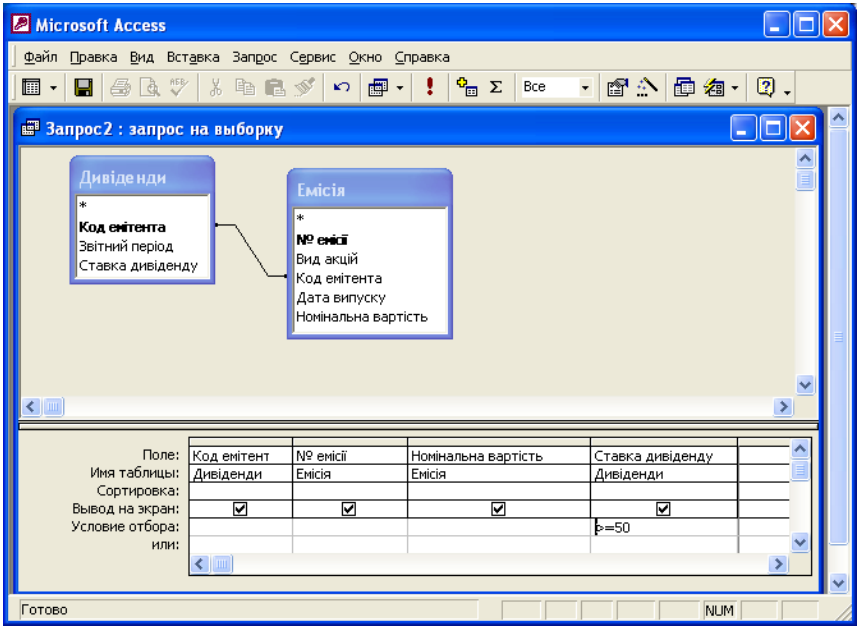
№ з/п	Вихідний об’єкт	Кінцевий об’єкт	Тип зв’язків
1.	Емітент	Дивіденди	1:Б
2.	Дивіденди	Емітент	Б:1
3.	Емітент	Емісія	1:Б
4.	Емісія	Емітент	Б:1
5.	Власник	Сертифікат	1:Б
6.	Сертифікат	Власник	Б:1
7.	Емітент	Сертифікат	1:Б
8.	Реєстратор	Сертифікат	1:Б
9.	Сертифікат	Реєстратор	Б:1
10.	Країна	Власник	1:Б
11.	Сертифікат	Емісія	Б:1
12.	Власник	Реєстратор	Б:1

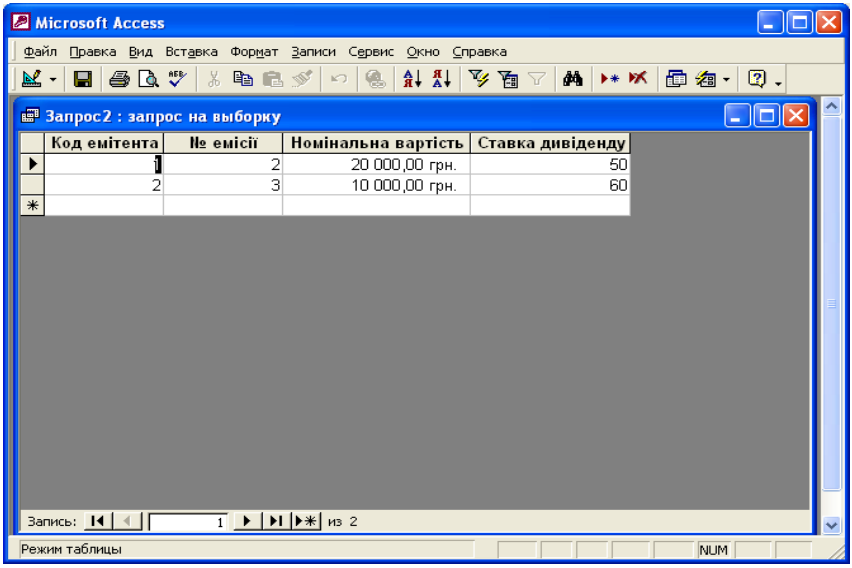
Приклад запиту згідно даних таблиці 2:





Вибір емітентів, ставка дивідендів яких більша або рівна 50 грн.:


Microsoft Access
Файл Правка Вид Вставка Запрос Сервис Окно Справка
Все
Запрос2 : запрос на выборку
Дивіденди
*
Код емітента
Звітний період
Ставка дивіденду
Емісія
*
№ емісії
Вид акцій
Код емітента
Дата випуску
Номінальна вартість
Поле: Код емітент № емісії Номінальна вартість Ставка дивіденду
Имя таблицы: Дивіденди Емісія Емісія Дивіденди
Сортировка:
Вывод на экран:
Условие отбора: >=50
или:
Готово
NUM


Microsoft Access
Файл Правка Вид Вставка Формат Записи Сервис Окно Справка
Запрос2 : запрос на выборку
Код емітента № емісії Номінальна вартість Ставка дивіденду
1 2 20 000,00 грн. 50
2 3 10 000,00 грн. 60
Запись: 1 из 2
Режим таблицы
NUM

РЕКОМЕНДОВАНА ЛІТЕРАТУРА

1. Уокенбах, Джон Excel 2003. Библия пользователя.: Пер. с англ. – М.: Издательский дом „Вильямс”, 2004. – 768 с. : ил.
2. Праг, Керри, Н., Ирвин, Майкл, Р., Access 2002. Библия пользователя.: Пер. с англ. – М.: Издательский дом „Вильямс”, 2004. – 1216 с. : ил.
3. Черняк О.І., Ставицький А.В., Чорноус Г.О. Системи обробки економічної інформації: Підручник. – К.: Знання, 2006. – 447 с. – (Вища освіта XXI століття).
4. В.М. Беспалов, А.Ю. Вакула, А.М. Гострик, С.Г. Діордіца, С.М. Таракановський, Є.В. Тиханович Інформатика для економістів: Навчальний посібник для студентів вищих навчальних закладів економічних спеціальностей. – К.: ЦУЛ, 2003. – 788 с.
5. Войтюшенко Н.М. Інформатика і комп’ютерна техніка: Навчальний посібник з базової підготовки для студентів економічних і технічних спеціальностей денної і заочної форм навчання / Н.М. Войтюшенко, А.І. Остапець. – К.: Центр навчальної літератури, 2006. – 568 с.
6. Клименко О.Ф., Головко Н.Р., Шарапов О.Д. Інформатика та комп’ютерна техніка: Навч.-метод. Посібник / За заг. ред. О.Д. Шарапова. – К.: КНЕУ, 2002. – 534 с.
7. Кишик А.Н. Эффективный самоучитель Excel 2003. – СПб.: ООО «ДиаСофтЮП»; СПб.: Питер, 2005. – 247 с.: ил.
8. Лабораторний практикум з інформатики та комп’ютерних технологій / В.В. Браткевич, І.О. Золотарьова, В.Є. Климнюк, І.П. Коврижних, В.П. Молчанов, О.М. Мокринський, В.І. Плоткін, О.І. Пушкар, В.В. Федько / За ред. О.І. Пушкаря: Навчальний посібник. – Х.: Видавничий дім «ІНЖЕК», 2003. – 424 с.
9. Ярмуш О.В., Редько М.М. Інформатика і комп’ютерна техніка: Навч. посібник. – К.: Вища освіта, 2006. – 359 с.: іл.
10. Золотарьова І.О. Табличний процесор Excel 2003 і його застосування в економіці. Навчальний посібник / І.О. Золотарьова, О.М. Мокринський; За ред. докт. екон. наук, професора О.І. Пушкаря. – Харків: Вид. ХНЕУ, 2005. – 188 с.
11. Кайт Т. Эффективное проектирование приложений Oracle. – М.: Изд. Лори, 2006. – 637с.
12. O’Brien J.A. Management information systems. – New York: Mc graw-hill, 2004. – 623с.
13. Ferrett R. Access 2000: Essentials Intermediate. – New York: Prentice Hall, 2000. – 342 с.
14. Haag S. Microsoft Excel 2002/ S.Haag, J.T.Perry. – New York: Mc Graw Hill, 2002. – 162с.

15. Grauer R. Access 2002: Exploring. Volume I/ R.Grauer, M.Barber. – New York: Prentice Hall, 2000. – 200 c.
16. Haag S. Microsoft Access 2002: Introductory/ S.Haag, J.Perry, M.Wells.- New York: McGraw-Hill, Inc, 2002. – 342 c. – (The I-Series)
17. Haag S. Microsoft Access 2002: Complete/ S.Haag, J.T.Perry, M.Wells.- New York: Mc Graw Hill, 2002. – 260 c.
18. Stephen M. Access 2000: Made Simple: Business edition. – New York: Prentice Hall, 2000. – 213 c.
19. Duffy T. Microsoft Excel 2002. – New York: Prentice Hall, 2002. – 208 c.
20. Stephen M. Access 2000: Made Simple: Business edition. – New York: Prentice Hall, 2000. – 213 c.
21. Duffy T. Microsoft Excel 2002. – New York: Prentice Hall, 2002. – 208 c.
22. Morrris S. Excel 2000: Made Simple: Business Edition. – London: Continuum, 2000. – 214 c.
23. Haag S. Management Inform Systems: For the information AGE. – London: Irwin, 2004. – 550 c.
24. Haag S. Microsoft Excel 2002: Complete/ S.Haag, J.Perry. – Boston: McGraw-Hill, Inc, 2002. – 630 c.
25. O'Brien J.A. Enterprise Information Systems/ J.A.O"Brien, G.M.Marakas.- New York: Mc Graw Hill, 2001. – 543 c.
26. Haag S. Management Information Systems: For the information AGE. – London: Irwin, 2004. – 550 c.
27. Preston J. Access 2000/ J.Preston, S.Preston, R.Ferrett. – New York: Prentice Hall, 2000. – 342 c.
28. O'Keefe T.L. Microsoft Excel 2000: Illustrated introductory Edition/ T.L.O'Keefe, E.E.Redi8ng. – New York: Course Technology, 2000. – 167c.
29. Preston J. Excel 2000/ J.Preston, S.Preston, R.Ferrett. – New York: Prentice Hall, 2000. – 293 c.
30. Management Information Systems: Mis Companion CD included. – 4-e.- New York: Course Technology, 2004. – 756c.
31. Management Information Systems: Preview Edition. – 4-e. – New York: Course Technology, 2004. – 756 c.
32. Friedrichsen L. Microsoft Access 2000: Illustrated Complete. – New York: Course Technology, 2000. – 514 c.
33. O'Brien J. Introduction to Information Systems. – 12-e. – New York: Mc Graw-Hill, 2005. – 476 c.

ЗМІСТ

Вступ 3

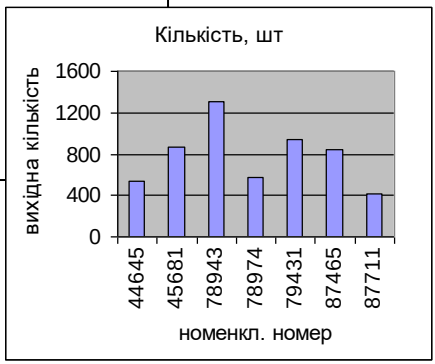
Розділ 1. Основи теорії баз даних 4

Розділ 2. Технологія роботи з базами даних в середовищі електронних таблиць.....18

Розділ 3. Технологія роботи з реляційними базами даних в середовищі СУБД ACCESS.....29

Приклади розв'язування задач створення і підтримки типових баз даних 48

Рекомендована література.....77



Підписано до друку хх.хх.2018 р.
Формат 60х84/16. Папір офсетний.
Друк на дублікаторі. Гарнітура Times. Зам. № х/хх-00
Умовн-друк. арк. 4,7. Облік.-видавн. арк. 4,8
Тираж ххх прим.