

Київський університет імені Бориса Грінченка
Інститут лідерства та соціальних наук
Кафедра інформатики

Абрамов В.О., Чегренець В.М.

ОСНОВИ БАЗ ДАНИХ ТА РОБОТА В СУБД ACCESS

Київ - 2013

*Рекомендовано
Вченою радою Інституту суспільства
Київського університету імені Бориса Грінченка
(протокол № 10 від 19 червня 2013 р.)*

УДК [510.51:004.9](075.8)

ББК 32.973.26-018.2

С 41

Рецензенти:

Жук С.Я. – доктор технічних наук, професор (Київський національний політехнічний університет).

Козлов В.В. – кандидат технічних наук, доцент, завідувач кафедри (Національна академія статистики, обліку та аудиту).

Абрамов В.О., Чегренець В.М. Основи баз даних та робота в СУБД Access: навчальний посібник для спеціальності «Інформатика». – К.: Київ. ун-т ім. Б. Грінченка, 2013. –100 с.

У навчальному посібнику надаються теоретичні відомості та практичні прийоми по проектуванню та створенню реляційних баз даних і додатків баз даних, як головної частини програмного забезпечення сучасних інформаційних систем. Питання розглядаються на прикладах системи управління базою даних Access 2007. Приведено етапи створення бази даних та основні принципи нормалізації. Розглянуто створення таблиць різної структури, установка зв'язків між таблицями, створення запитів. Значна увага приділена в посібнику створенню різних по призначенню та структурі форм, порядку налагодження форм, побудові та модифікації звітів. Пропонується перелік контрольних питань після матеріалу кожного розділу посібника.

Навчальний посібник підготовлено відповідно до курсу «Бази даних та інформаційні системи», має на меті допомогти формуванню фундаментальних теоретичних знань і практичних навичок з проектування баз і банків даних, їх створення та ведення із застосуванням засобів сучасних СУБД Microsoft Access 2007. Призначений для студентів і викладачів вищих педагогічних навчальних закладів освіти.

ЗМІСТ

ВСТУП	5
1. Введення до баз даних	8
1.1. Основні поняття інформаційних систем	8
1.2. Моделі даних.....	9
1.3. Структура таблиць.....	11
1.4. Ключі та зв'язки.....	12
1.5. Система управління базою даних	14
1.6. Контрольні питання до 1 розділу.....	16
2. Проектування і створення БД і додатків	16
2.1. Етапи проектування БД.....	16
2.2. Принципи нормалізації	19
2.3. Етапи створення БД.....	21
2.4. Контрольні питання до 2 розділу	22
3. Робота з СУБД Microsoft Access	22
3.1. Запуск Microsoft Access	22
3.2. Функції СУБД Microsoft Access	23
3.3. Склад інтерфейсу Microsoft Access 2007.....	25
3.4. Контрольні питання до 3 розділу	26
4. Розробка таблиць у СУБД Microsoft Access.....	26
4.1. Створення таблиці	27
4.2. Ключові поля та індекси	36
4.3. Установка зв'язків між таблицями в СУБД Access.....	39
4.4. Коректування таблиці	42
4.5. Операції з таблицями	44
4.6. Контрольні питання до 4 розділу	45
5. Робота з даними у таблицях	45
5.1. Відкривання таблиці та навігація.....	45
5.2. Введення, редагування і видалення записів.....	46
5.3. Пошук і заміна даних у таблицях.....	49
5.4. Сортування і фільтрація записів	53
5.5. Контрольні питання до 5 розділу	55
6. Створення запитів.....	55
6.1. Створення запиту на вибірку.....	56
6.2. Сортування і умови відбору даних	58
6.3. Обробка даних у запиті.....	62
6.4. Мова запитів SQL	64
6.5. Контрольні питання до 6 розділу	66
7. Форми та робота з ними.....	67

7.1.	Створення форми.....	68
7.2.	Конструктор форм.....	69
7.3.	Налагодження форми.....	72
7.4.	Типи полів.....	75
7.5.	Складні форми.....	80
7.6.	Контрольні питання до 7 розділу.....	83
8.	Створення і використання звітів.....	83
8.1.	Типи звітів.....	84
8.2.	Створення звітів.....	86
8.3.	Структура звітів.....	87
8.4.	Обчислення у звітах.....	91
8.5.	Перегляд звіту.....	94
8.6.	Контрольні питання до 8 розділу.....	95
9.	Створення макросів.....	95
9.1.	Поняття макросу.....	95
9.2.	Створення макросів.....	96
9.3.	Макроси з умовними макрокомандами.....	99
9.4.	Макроси у формах.....	100
9.5.	Контрольні питання до 9 розділу.....	104
10.	Додатки.....	104
10.1.	Додаток 1. Створення шаблону пошуку.....	104
	Література:.....	106

ВСТУП

XXI століття в усіх сферах життєдіяльності людини і суспільства в цілому висуває нові, раніше невідомі завдання. Це ставить перед людиною, а отже, і перед освітою, нові завдання, зумовлені переходом людства на перетині тисячоріч до нового типу цивілізації.

«Розбудова системи освіти, її докорінне реформування, — наголошується в національній програмі «Освіта» (Україна XXI століття),— мають стати основою відтворення інтелектуального, духовного потенціалу народу, виходу вітчизняної науки, техніки і культури на світовий рівень, національне відродження, становлення державності та демократизації суспільства в Україні».

Освіта XXI століття – це освіта для людини, відповідальної особистості, яка здатна до самоосвіти і саморозвитку, вміє критично мислити, опрацьовувати різноманітну інформацію, використовувати набуті знання і вміння для творчого розв’язання проблем, прагне змінити на краще своє життя і життя своєї країни.

XXI століття – це час переходу до високотехнологічного інформаційного суспільства, у якому якість людського потенціалу, рівень освіченості і культури всього населення набувають вирішального значення для економічного і соціального поступу країни.

Сучасна освіта направлена на формування цілісної інформаційної культури майбутнього громадянина інформаційного суспільства. Рішення цієї задачі можливе лише тоді, коли в освіту прийде вчитель інформатики, який сам володіє цілісною інформаційною культурою, інформаційним світоглядом і здатний сформувати їх у своїх учнів. Тому формування інформаційного світогляду вчителя інформатики – одне з пріоритетних завдань сучасної педагогічної освіти.

У навчальному посібнику надаються теоретичні відомості та практичні прийоми по проектуванню та створенню реляційних баз даних і додатків баз даних, як головної частини програмного забезпечення сучасних інформаційних систем. Питання розглядаються на прикладах системи управління базою даних Access 2007. Приведено етапи створення бази даних та основні принципи нормалізації. Розглянуто створення таблиць різної структури, установка зв’язків між таблицями, створення запитів. Значна увага приділена в посібнику створенню різних по призначенню та структурі форм, порядку налагодження форм, побудові та модифікації звітів. Пропонується перелік контрольних питань після матеріалу кожного розділу посібника.

Навчальний посібник містить в собі дев’ять розділів по основах

баз даних та роботі в СУБД Access:

- бази даних та інформаційні системи, моделі даних, реляційні бази даних; в ньому проводиться аналіз структур таблиць з послідовним переходом до системи управління базою даних;

- створення БД і додатків, проектування БД, аналіз сутностей їх атрибутів, визначення зв'язків між базами даних, принципи нормалізації та етапи створення БД;

- робота з СУБД Microsoft Access, порядок запуску Microsoft Access, функції СУБД Microsoft Access, склад інтерфейсу Microsoft Access 2007;

- розробка таблиць у СУБД Microsoft Access, створення таблиці, пояснення понять ключові поля та індекси, створення структури таблиць, установка зв'язків між таблицями в СУБД Access, коректування таблиці, операції з таблицями;

- робота з даними у таблицях, відкривання таблиці та навігація, введення, редагування і видалення записів, пошук і заміна даних у таблицях, сортування і фільтрація записів, створення запитів, створення запиту на вибірку, сортування і умови відбору даних, обробка даних у запиті;

- форми та робота з ними, створення форм, налагодження форми, складні форми;

- створення і використання звітів, типи звітів, структура звітів, елементи керування, створення звітів, обчислення у звітах, перегляд звіту;

- створення макросів, поняття макросу, створення простих макросів, макроси з умовними макрокомандами, макроси у формах.

Всі розділи та додаток доцільно використовувати для проведення практичних та лабораторних занять за різноманітними варіантами завдань студентам навчальних груп по дисциплінах «Математичні методи та комп'ютерні засоби в дослідженнях», «Бази даних та інформаційні системи», «Системи управління базами даних» та ін.

Навчальний посібник підготовлено відповідно до курсу «Бази даних та інформаційні системи», має на меті допомогти формуванню фундаментальних теоретичних знань і практичних навичок з проектування баз і банків даних, їх створення та ведення із застосуванням засобів сучасних СУБД Microsoft Access 2007. Посібник призначений для студентів і викладачів вищих педагогічних навчальних закладів освіти.

У посібнику надаються приклади і зображення вікон з додатку Access 2007. У тексті жирним шрифтом позначені назви об'єктів, які

з'являються на екрані (назви команд, вікон і т. ін.) . Подвійним підкресленням позначаються назви пунктів меню, а однократним підкресленням – розділи панелі інструментів. Складові шляхів до команд відокремлюються знаком «/». Наприклад для вирівнювання тексту по центру виконується команда: Главная / Абзац / **По центру**. При необхідності у дужках даються переклади назв об'єктів і команд для російськомовних версій додатку Access 2007, якщо вони суттєво відрізняються від україномовних.

1. Введення до баз даних

1.1. Основні поняття інформаційних систем

База даних це поійменована і організована відповідно до певних правил сукупність даних, що забезпечує їх збереження, оновлення та маніпулювання у визначеній предметній області та використовується для задоволення інформаційних потреб користувачів із застосуванням електронної обчислювальної техніки.

Предметна область - це частина реального світу, що підлягає вивченню і управлінню. Для автоматизації цих процесів створюються бази даних.

Розрізняють терміни «база даних» і «система управління базами даних (СУБД)». База даних це безпосередньо дані, які зберігаються, а СУБД це сукупність програмних і алгоритмічних засобів загального або спеціального призначення, які забезпечують управління створенням і використанням баз даних.

Безпосередньо використання даних і вирішення конкретних прикладних завдань здійснюють **додатки баз даних**. Додатки створюються для конкретного користувача під конкретну задачу. Вони містять запити, форми, звіти, які пристосовані для виконання конкретних прикладних задач.

База даних є однією із складових інформаційної та інформаційно-аналітичної системи органу управління, її інформаційно-довідковим ядром. **Інформаційна система** – це сукупність людського інтелекту, інформаційних ресурсів, сучасних інформаційних технологій та засобів інформатизації, об'єднаних у телекомунікаційні і комп'ютерні мережі. Вона створюється з метою автоматизації процесу збору й обробки інформації, її всебічної оцінки в реальному масштабі часу, проведення системного аналізу отриманої інформації та обґрунтування рішень, що відпрацьовуються в органах управління для вирішення поставлених задач.

Одне з основних застосувань баз даних це програмне забезпечення інформаційних систем. Бази даних є сучасною формою організації програмних засобів для зберігання і доступу до інформації в інформаційних системах. Прикладами великих інформаційних систем є банківські системи, системи заказу залізничних квитків і т. ін. В навчальному посібнику розглядаються тільки програмні засоби інформаційних систем.

Банк даних - автоматизована інформаційна система централізованого зберігання и колективного використання даних. В состав

банка входять одна або кілька баз даних, довідник баз даних, СУБД, а також бібліотеки запитів и прикладних програм додатків БД. Дуже часто термін база даних розуміють як банк даних. Ці терміни майже схожі, тому треба уважно розглядати про що йдеться мова.

1.2. Моделі даних.

Набори принципів, які визначають організацію логічної структури зберігання даних в базі, називаються **моделями даних**. До базових понять моделі БД відносяться: сутності, зв'язки між ними і їх атрибути (властивості).

Сутність - будь-який конкретний або абстрактний об'єкт в предметній області. Сутності - це базові типи інформації, які зберігаються в БД (в реляційній БД кожній сутності призначається таблиця).

До сутності можуть належати: студенти, клієнти, підрозділи і т.д. Примірник (реалізація) сутності (об'єкта) і тип сутності - це різні поняття. Поняття тип сутності відноситься до набору однорідних особистостей, предметів або подій, виступаючих як ціле (наприклад, студент, клієнт і т.д.). Типу сутності відповідає таблиця у БД. Примірник сутності відноситься, наприклад, до конкретної реалізації особистості в наборі. Примірнику сутності відповідає рядок у таблиці БД. Типом сутності може бути студент, а примірником - Петров, Сидоров і т. д.

Атрибут - це властивість сутності у предметній області. Його найменування має бути унікальним для конкретного типу суті. Атрибут - це логічно неподільний елемент, що належить до властивості деякого об'єкта чи процесу. Наприклад, для сутності Студент можуть бути використані наступні атрибути: прізвище, ім'я, по батькові, дата і місце народження, паспортні дані і т.д. У БД атрибути зберігаються в полях таблиць.

Зв'язок - взаємозв'язок між сутностями в предметній області. Зв'язки являють собою з'єднання між частинами БД (в реляційній БД - це з'єднання між записами таблиць).

Сутність - це дані, які класифікуються за типом, а зв'язки показують, як ці типи даних співвідносяться один з одним. Якщо описати деяку предметну область у термінах суті - зв'язок, то отримаємо модель сутність - зв'язок для цієї БД.

Усі бази, незалежно від їх конкретного вмісту, можуть бути класифіковані за певними ознаками: за предметною областю, за змістом, за технологією зберігання даних, за ступенем розподілу і т. д. Зокре-

ма, за моделлю даних виділяють ієрархічні, мереживі, реляційні та інші типи бази даних.

Ієрархічні бази даних можна уявити як дерево, що складається з об'єктів різних рівнів. Перший, верхній рівень займає один об'єкт, другий – об'єкти другого рівня і т. д. Між об'єктами існують зв'язки, кожний об'єкт може включати до себе декілька об'єктів нижчого рівня. Пошук даних здійснюється послідовно зверху вниз за рівнями відповідно до існуючих зв'язків між ними.

Мереживі бази даних подібні ієрархічним за виключенням того, що вони мають покажчики в обох напрямках, які з'єднують споріднену інформацію. У таких базах використовуються графові форми подання даних. Пошук даних можна здійснювати з будь-якого рівня послідовним доступом до інформаційних масивів згідно з встановленими між ними зв'язками (за так званими адресними посиланнями).

В **реляційних** базах даних (Relational Database System, RDBS) всі дані відтворюються у двовірних таблицях. Таблиці відображають відношення (relation), які існують між даними. Таким чином, база даних є не що інше, як множина таблиць. Пошук потрібних даних спрощується до визначення назви таблиці, рядка і стовпця, які і визначають адресу потрібного даного з бази. Сукупність таких поименованих таблиць і створює основу єдиної бази даних.

Реляційна база даних позбавлена недоліків, притаманних двом попереднім типам, та краще відповідає всім основним вимогам, що висуваються до баз даних, потрібних для організації повсякденної діяльності інформаційної системи:

- забезпечення можливості одночасного використання даних в різних прикладних програмах різними користувачами;
- забезпечення можливості багаторазового використання даних прикладними програмами та накопичення їх за результатами рішення задач;
- незалежність даних від прикладних програм;
- виключення надмірності даних та їх дублювання;
- простота звернення до бази даних, зміни її логічної і фізичної структури та розвитку;
- цілісність, захист та таємність баз даних.

Тому надалі будемо розглядати саме реляційні бази даних.

1.3. Структура таблиць.

Предметна область складається з об'єктів (сутностей) і їх властивостей, які пов'язані один з одним. Структура БД відповідає об'єктам предметної області. Властивості об'єктів можуть бути охарактеризовані за допомогою сукупності кількісних та якісних ознак – атрибутів. Якість визначають атрибути-прикмети, а кількість – атрибути-основи.

Атрибути-прикмети є якісною характеристикою об'єкта і застосовуються в логічних операціях, таких як порівняння, сортування, компонування та редагування. Атрибути-основи характеризують кількісні якості об'єкта, пов'язані з атрибутами-прикметами та застосовуються в обчислювальних операціях. Атрибути-основи, наприклад, кількість, дальність, час, без атрибутів-прикмет не дають визначення об'єкта.

Для запису значень атрибутів у базі даних відводяться поля (стовпці) таблиці. Поле – це найменша кількість символів, що визначає властивості об'єкта, тобто це – найменша пойменована одиниця даних.

Сукупність полів (сукупність атрибутів) конкретної реалізації об'єкта називається запис (рядок, кортеж). Записи в базі даних містять відомості про окремі реалізації об'єктів та розташовуються у рядках пойменованої таблиці. Таким чином кожному об'єкту відповідає своя таблиця. Кожній реалізації об'єкта відповідає запис у таблиці. Прикладом об'єкту є людина, її атрибути – прізвище, ім'я, по батькові, рік народження і т. ін. Наприклад, одна з реалізацій об'єкта це Іванов Іван Іванович, друга Петров Петро Петрович і т.і

Кожний запис у таблиці повинен мати хоча б одну унікальну ознаку - ключ, за якою він може бути однозначно виділений (ідентифікований) серед інших. Для запису цієї ознаки виділяється спеціальне поле або кілька полів. Кількість ключових полів залежить від інформації у полі і кількості записів у таблиці.

Наприклад, унікальний ключ для людини може бути її прізвище. Але при більшій кількості записів можливі однофамільці, тому у ключові поля слід включити ім'я, а інколи ще і по батькові і рік народження і т.і. Часто дійсно унікальними є коди, наприклад, ідентифікаційний код. У Access традиційно використовують ключ типа лічильник, який автоматично збільшується для кожного нового запису і не повторюється навіть при видаленні запису. Приклад таблиці зображено на рис. 1 .

Рядок, запис	Код	Прізвище	Імя	По батькові	Рік народж.
	1	Іванов	Іван	Іванович	1995
	2	Петров	Петро	Петрович	1994
	3				

Ключ Атрибут Атрибут Атрибут Атрибут

Рис. 1 Приклад таблиці і її елементів.

1.4. Ключі та зв'язки.

Дані про об'єкти в базах даних узгоджені між собою зв'язками. Зв'язки між записами, що розташовані у різних таблицях, реалізуються за допомогою ключів. Ключ – це певним способом помічені стовпці (поля) таблиці, за допомогою яких ідентифікуються і розрізняються реалізації об'єктів, а також здійснюється зв'язок між таблицями. Цим досягається раціональний розподіл даних у таблицях і виключається надмірність таблиць.

Ключовим полем може бути практично будь-яке поле в таблиці, оскільки воно визначається розробником бази. Ключ - це стовпчик (або кілька стовпців таблиці дозволяє встановити зв'язок із записами в іншій таблиці. Існують ключі двох типів первинні, вторинні або зовнішні.

Первинний ключ - це одне або декілька полів (стовпчиків), комбінація значень яких однозначно визначає кожний унікальний запис у таблиці. Первинний ключ не допускає порожнього значення (значення Null) і завжди повинен мати унікальний індекс. Первинний ключ використовується для зв'язку таблиці з зовнішніми ключами в інших таблицях.

Зовнішній (вторинний) ключ - це одне або декілька полів (стовпців) у таблиці, що містять посилання на поля первинного ключа в іншій таблиці. Зовнішній ключ визначає спосіб об'єднання таблиць.

Зв'язок встановлюється між двома спільними полями (стовпцями) двох таблиць. Полями у яких дані однакові. Відносини, які можуть існувати між записами двох таблиць:

1. Один – до - одного, кожному запису з однієї таблиці відповідає один запис в іншій таблиці. Відношення «один до одного» ство-

рюється в тому випадку, коли обидва поля є ключовими або мають унікальні індекси.

2. Один - до - багатьом, кожного запису з однієї таблиці відповідає кілька записів іншій таблиці. Відношення «один-до-багатьом» створюється в тому разі, коли одне з полів є полем первинного ключа або унікального індексу.

3. Багато – до - одному, безлічі записів з однієї таблиці відповідає один запис в іншій таблиці;

4. Багато - до - багатьом, безлічі записів з однієї таблиці відповідають кілька записів в іншій таблиці. Відношення «багато-до-багатьом» фактично є двома відношеннями «один-до-багатьом» базових таблиць з третьою таблицею, первинний ключ якої складається з полів вторинного ключа двох базових таблиць.

З двох логічно пов'язаних таблиць одну називають таблицею первинного ключа або головної таблицею, а іншу таблицею вторинного (зовнішнього) ключа або підлеглою таблицею. СУБД дозволяють зіставити записи з двох таблиць і спільно вивести їх у звіті або запиті. На рис. зображений приклад зв'язку один - до - багатьом між таблицями Академгрупа і Студенти.

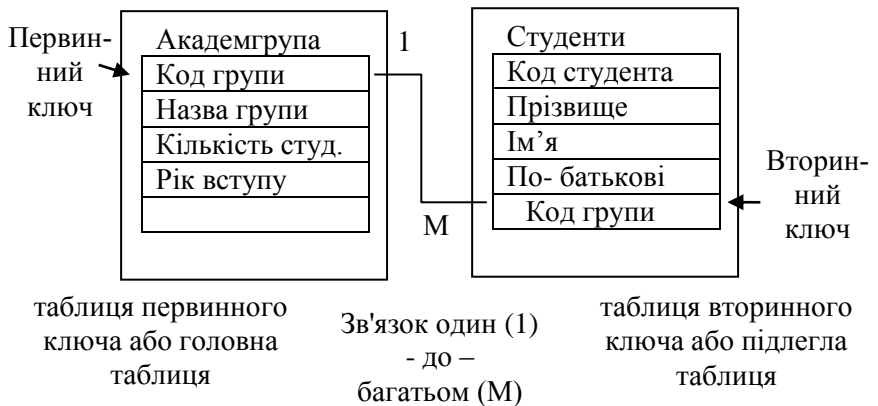


Рис. 2 Приклад зв'язку один - до - багатьом між таблицями.

Існує три типи первинних ключів: ключові поля лічильника (лічильник), простий ключ, складний ключ.

Поле **лічильника** (тип даних «Лічильник»). Тип даних поля в базі даних, в якому для кожного рядку, що додається в таблицю, автоматично заноситься унікальне числове значення.

Простий ключ. Якщо поле містить унікальні значення, такі як коди або інвентарні номери, то це поле можна визначити як первинний ключ. В якості ключа можна визначити будь-яке поле, що містить дані, які не повторюються і не мають пусте значення Null.

Складний ключ. У випадках, коли неможливо гарантувати унікальність значень кожного поля, існує можливість створити ключ, складений з декількох полів. Найчастіше така ситуація виникає для таблиці, використовуваної для зв'язування двох таблиць багато - до - багатьом.

Необхідно ще раз відзначити, що в полі первинного ключа повинно бути лише унікальні значення в кожному рядку таблиці, тобто збіг не допускається, а в полі вторинного ключа збіг значення у рядках таблиці допускається. Якщо виникають труднощі з вибором відповідного типу первинного ключа, то в якості ключа доцільно вибрати поле лічильника.

Унікальність має бути як зараз так і у майбутньому. Наприклад, БД у фірми, в якій всього три співробітника - Іванов, Петров і Сидоров. Можна зробити ключем прізвище, але виникають проблеми, якщо фірма розшириться і на роботу прийде однофамілець наявного співробітника. Розширення ключа на ще два стовпчики - з іменами і по батькові - також не є вирішення проблеми. Може з'явитися повний тезко і однофамілець. Крім того, якщо, приміром, співробітниця змінить прізвище, то знову виникають проблеми - вона буде виглядати для бази даних як новий співробітник. На практиці зазвичай створюється спеціальне поле (найчастіше - Integer з властивістю Identity в SQL Server або стовпчик лічильника в Access), який унікально ідентифікує запис. Таке рішення володіє рядом переваг з точки зору продуктивності і зберігання даних.

1.5. Система управління базою даних

Чітка структура реляційних баз даних дозволяє достатньо просто здійснити автоматизацію їх обробки. Для цього створюються спеціальні програми (точніше, комплекси програм), призначенням яких є організація та ведення баз даних. Такі програми називаються системами управління базами даних (СУБД). СУБД призначені для структурування інформації, розміщення її в таблицях і маніпулювання даними. Іншими словами СУБД призначені як для створення і ведення бази даних, так і для доступу до даних. В даний час налічується біль-

ше 50 типів СУБД для персональних комп'ютерів. До найбільш поширених типів СУБД відносяться: MS SQL Server, Oracle, Informix, Sybase, DB2, MS Access і т. ін.

СУБД забезпечує взаємодію користувача і прикладних програм з базами даних, а також виконання спеціальних функцій в середовищі бази даних:

- створення бази даних;
- ведення бази даних, тобто оперативну зміну даних, корегування параметрів та розширення бази;
- пошук та завантаження бази даних;
- визначення порядку доступу до даних та їх обробки;
- визначення порядку добування даних з бази з метою вирішення розрахункових задач та отримання інформаційних довідок;
- друкування даних, ведення обліку використання даних, їх захист та відновлення тощо.

Поєднання бази даних та системи управління нею утворює банк даних. Банк даних – це комплекс програмних, мовних, організаційних та технічних засобів, призначених для централізованого накопичення та колективного використання даних, а також самі дані, що зберігаються в базах даних.

База даних повинна бути зручною для конкретної мети використання даних предметної області. Тому для кожної мети засобами системи управління базами даних створюється спеціальний додаток бази даних. Це спеціалізовані форми, звіти, запити, які задовольняють поставленій меті. Таблиці баз даних і додатки баз даних рідко розділяють і використовують окремо, частіше вони поєднуються у понятті **База даних**.

Одною з головних вимог до будь-якої бази даних є точна її відповідність конкретній обстановці, що склалася на цей час. Оскільки обстановка постійно змінюється, це означає, що база даних є динамічним об'єктом. Контроль за наявними даними, перевірка їх достовірності, безперервне уточнення, поповнення записів та вилучення непотрібних є предметом постійної уваги адміністратора. Більш того, можна стверджувати, що зміна конкретної обстановки може привести до необхідності зміни структури бази, її таблиць та наявних полів.

у зв'язку з розвитком можливостей апаратних та програмних засобів все більш поширеним стає поняття **сховище даних** (Data Warehouse). Така назва відноситься до дуже великої предметно-орієнтованої інформаційної корпоративної бази даних, що спеціально розробляється для підтримки прийняття рішень. Сховище даних буду-

ється на основі клієнт-серверної архітектури, реляційної бази даних та утиліт підтримки прийняття рішень.

1.6. Контрольні питання до 1 розділу.

1. Що таке база даних?
2. Що таке і які бувають ключові поля?
3. Що таке банк даних?
4. Що таке сутність, атрибут?
5. Що таке і які бувають зв'язки ?
6. Яким чином можна створити файл нової бази даних?
7. Що таке структура таблиці?
8. Якими способами можна створити таблицю?
9. Що таке СУБД? Як класифікуються і використовуються СУБД?
10. Функції СУБД.
11. Дати аналіз баз даних, які створюються та використовуються у практичній діяльності організацій та окремих людей.
12. Класифікації баз даних за певними різними ознаками.
13. Що таке ієрархічні бази даних та їх характеристики?
14. Що таке мережеві бази даних та їх характеристики?
15. Що таке реляційна база даних та їх характеристики?

2. Проектування і створення БД і додатків

Основні задачі проектування:

1. Забезпечення зберігання в БД всієї необхідної інформації.
2. Забезпечення можливості отримання даних по усім необхідним запитам.
3. Скорочення надлишковості та дублювання даних.
4. Забезпечення цілісності бази даних.

2.1. Етапи проектування БД

Створення БД починається з проектування. Безпосередньо робота з програмою СУБД починається після здійснення всіх підготовчих робіт на папері. На це слід звернути увагу: чим ретельніше продумана, спланована база на папері, тим продуктивнішою буде робота з її створення у середовищі СУБД. Етапи проектування БД:

- Дослідження предметної області;
- Аналіз даних (сутностей, їх атрибутів);

- Визначення стосунків між сутностями і визначення первинних і вторинних (зовнішніх) ключів;

- Нормалізація.

В процесі проектування визначається структура реляційної БД (склад таблиць, їх структура і логічні зв'язки). Структура таблиці визначається складом стовпчиків, типом даних і розмірів стовпців, ключами таблиці.

Усі дані можна класифікувати за їх приналежністю, важливістю, способом подання, часом зміни, частотою звернення до них і т.д. Відповідно треба сформувати базу даних: визначити розподіл даних по таблицях, структуру таблиць, встановлення зв'язків між ними, порядок звернення до баз даних і т.д.

При дослідженні предметної області роботу із створення бази даних доцільно виконувати у такому порядку:

- формулювання назви і мети створення бази даних;

- точне визначення завдань відповідно до виконання службових обов'язків і формулювання завдань, які треба виконувати з допомогою бази даних;

- перелік даних, що потрібні для виконання цих завдань;

- розподіл даних за об'єктами (по розділах);

- визначення зв'язків між розділами;

- створення переліку які треба створити таблиці, форми, запити, звіти.

Розглянемо предметну область, яка пов'язана з навчанням студентів. БД «Успішність навчання» має мету – контроль за відмітками студентів з різних дисциплін і надання відповідної інформації:

- знати відмітки кожного студента з кожній дисципліні;

- мати загальний перелік академгруп і студентів, дисциплін і викладачів.

У БД повинні зберігатися дані про студентів, групах студентів, про оцінки студентів з різних дисциплін, про викладачів, про стипендії і т.д. Мінімальна база має дані про студентів, групи студентів і про оцінки студентів з різних дисциплін. Визначимо сутності, атрибути сутностей та основні вимоги до функцій БД.

Основними значущими сутностями цієї БД являються: Студенти, Групи студентів, Дисципліни, Успішність.

Основні предметно значущі атрибути сутностей:

- Студенти - прізвище, ім'я, по батькові, стать, дата і місце народження, академгрупа;

- Академгрупа - назва, курс, семестр;

- Дисципліни - назва, викладач, кількість годин;
- Успішність – студент, дисципліна, оцінка, вид контролю, дата контролю.

Основні вимоги до функцій БД:

- представити успішність студента з дисциплін із зазначенням загальної кількості годин і види контролю;
- представити успішність студентів за групами і дисциплінами;
- представити дисципліни, які вивчаються групою студентів на визначеному курсі або визначеному семестрі.

З аналізу даних предметної області випливає, що кожній сутності необхідно призначити найпростішу таблицю. Далі необхідно встановити логічні зв'язки між таблицями. Між таблицями Студенти і Успішність необхідно встановити такий зв'язок, щоб кожного запису таблиці Студенти відповідало кілька записів у таблиці Успішність, тобто один - до - багатьом, оскільки у кожного студента може бути кілька оцінок.

Логічний зв'язок між сутностями Групи - Студенти визначена як один - до - багатьом виходячи з того, що в групі є багато студентів, а кожен студент входить до складу однієї групи. Логічний зв'язок між сутностями Дисципліни - Успішність визначена як один - до - багатьом, тому що з кожної дисципліни може бути поставлено кілька оцінок різних студентам.

На підставі вищевикладеного складаємо модель сутність - зв'язок для БД «Успішність навчання» (рис. 3).

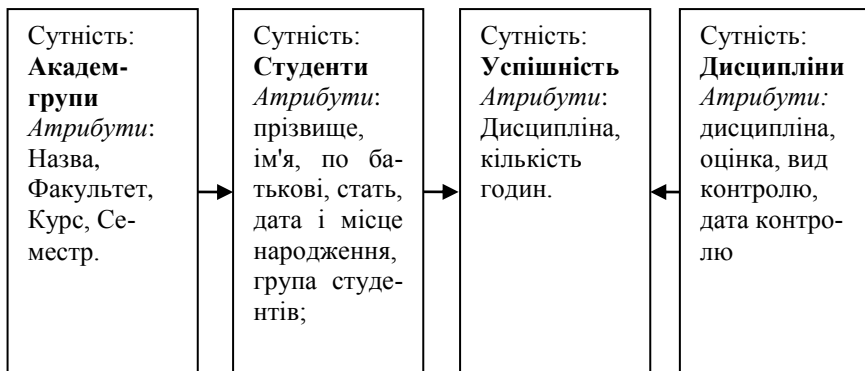


Рис.3 Модель бази даних **Успішність навчання** (стрілка є умовними позначеннями зв'язку: один → до → багатьом).

Перелік вирішуваних завдань завжди можна розподілити на окремі складові за тематикою, часом звернення до них, порядком оновлення даних. Відповідно до цього слід формувати розділи бази даних.

Після формування розділів бази даних необхідно скласти детальний перелік всіх даних, що мають використовуватися під час вирішення конкретних задач, передбачаючи при цьому джерела їх одержання та порядок введення до відповідних розділів. В процесі роботи з базою даних дані можуть додаватися та вилучатися.

Зрозуміло, що сформувати повний перелік всіх даних під час розробки структури бази даних не завжди можливо. Тому слід враховувати, що завжди можна внести зміни у структуру, якщо під час подальшої роботи ці зміни стануть очевидними.

Заповнення створених таблиць даними є одним з перших та відповідальних етапів розробки бази даних. Від якості виконання цього етапу суттєво залежить ефективність використання даних під час підготовки та обґрунтування рішення, оскільки таблиці складають основу будь-якої реляційної бази даних. Саме тут зберігаються всі потрібні суб'єкти, які задаються множиною своїх ознак (властивостей).

Від якості проектування таблиць (визначення полів, типів та властивостей даних) залежить якість використання даних, що зберігаються в них.

Важливим моментом при створенні багатотабличних баз даних є визначення зв'язків між таблицями. Від вірного вирішення цього питання залежать можливості та зручність користування базами у повному обсязі: створення запитів та форм з виконанням корегування записів та внесенням нових записів.

Зв'язки між таблицями здійснюються за допомогою спеціальних ключових полів. Кожна таблиця обов'язково повинна мати первинний ключ: одне чи декілька полів, що мають унікальні для кожного запису значення. Не може бути хоча б двох записів з однаковими ключами. Якщо з таблицею треба зв'язати іншу, у останній мають бути передбачені поля чужого ключа – первинного ключа першої таблиці.

2.2. Принципи нормалізації

Створення БД починається із створення таблиць. При цьому слід користуватися спеціальними принципами. Нормалізація призначена для приведення структури БД до вигляду, що забезпечує мінімальну логічну надмірність, і не має на меті зменшення або збільшення продуктивності роботи чи фізичного обсягу бази даних. Кінцевою метою

нормалізації є зменшення потенційної суперечливості інформації. Загальне призначення процесу нормалізації полягає в наступному:

- виключення деяких типів надмірності;
- усунення деяких аномалій оновлення;
- розробка проекту бази даних, який є досить якісним» поданням реального світу, інтуїтивно зрозумілий і може служити гарною основою для подальшого розширення;
- спрощення процедури застосування необхідних обмежень цілісності.

Усунення надмірності проводиться, як правило, за рахунок декомпозиції відносин таким чином, щоб у кожному відношенні зберігалися лише первинні факти (тобто факти, які не можливо отримати з інших фактів).

Перш ніж нормалізувати дані, необхідно розглянути принципи, які лежать в основі нормалізації:

Перший принцип - декомпозиція без втрат. Це означає, що після розбивки ненормалізованої таблиці на декілька більш дрібних її можна за бажання об'єднати назад без втрати даних. Таке об'єднання зазвичай проводиться, не на рівні самої бази даних, а на рівні запитів.

Другий принцип - кожен кортеж (запис) повинен бути унікальним, тобто має бути можливість відрізнити один запис від іншого. Кожен запис має містити в собі мітку, це відрізняє її від інших записів. Така позначка називається ключем. Можна дати і більш формальне визначення ключа: ключ - це набір стовпців таблиці, значення яких унікально визначають рядок. Також будь-яке поле таблиці повинне надавати унікальний тип інформації; У кожній таблиці БД не повинно бути повторюваних полів;

Третій принцип - принцип функціональної залежності. Це означає, що для будь-якого кортежу існує якийсь набір атрибутів, унікальний для кожного кортежу цього відношення, і, знаючи цей набір, можна визначити значення інших, неунікальних атрибутів. При цьому має виконуватися принцип незалежності полів: зміни даних будь-якого поля не повинні впливати на дані інших полів.

У нормалізованій базі даних всі атрибути будуть функціонально залежати від ключів. Якщо ця умова не виконується, то процес нормалізації не завершено.

Після проектування починається безпосередньо створення бази даних.

2.3. Етапи створення БД

Після проектування починається створення БД (таблиці) і додатку БД (форми, запити, звіти) засобами СУБД. Перед цим вирішуються деякі питання:

Вибір потрібних даних з бази та їх оформлення.

Одержання потрібних даних з бази, які можуть бути розташовані у різних таблицях, виконується за допомогою запитів. Залежно від конкретної потреби запити можуть створюватися на вибірку, відновлення, видалення, додавання. При цьому слід враховувати, що запити дозволяють виконувати певні зміни у базі. Запити дозволяють також створювати нові таблиці з використанням даних з однієї чи декількох таблиць.

При роботі над створенням запиту основним є визначення умов на відбір даних, які формуються так же, як і умови на значення у таблицях.

Створення та зміни таблиць, запитів, форм бази даних.

Доцільність та спрямованість змін виявляється в процесі користування базами та виявлення невідповідності їх задуму або незручності користування. СУБД не накладає якихось обмежень на здійснення змін у базі. Вона забезпечує повну переробку об'єктів, включаючи зміну ключових полів.

Організація ведення баз даних.

Успішне користування даними, що знаходяться у базі, вимагає їх повної відповідності обстановці, що склалася. Це означає, що необхідне безперервне та своєчасне оновлення даних: додавання нових записів, зміну значень у полях таблиць, видалення зайвих записів і т.д.

Порядок внесення змін у базу даних, точніше у її таблиці, визначає користувач залежно від характеру потрібних змін. Слід мати на увазі, що всі об'єкти бази: таблиці, запити, форми є взаємозв'язаними. Тому внесення змін в один об'єкт веде до відповідної зміни даних у інших.

Робота із внесення змін у базу, зрозуміло, може виконуватися лише посадовцями, які наділені відповідними правами та несуть відповідальність за виконання робіт. З цього випливає, що одним з завдань, яке обов'язково має вирішувати начальник штабу (органу управління), є визначення порядку доступу до баз даних посадових осіб штабу. Взагалі заходи щодо збереження баз даних від несанкціонованого доступу повинні бути у центрі уваги керівників органу управління будь-якого рівня.

MS Access надає максимальних зручностей виконавцеві під час роботи на кожному з перелічених етапів. СУБД має розвинену систему команд для самостійного створення бази даних та множини програм-майстрів, що автоматизують роботу користувача. До того ж слід враховувати, що вона входить до комплексу офісних програм фірми Microsoft. Тому робота з вікнами, меню, панелями інструментів здійснюється за загальними правилами і не потребує додаткових пояснень.

2.4. Контрольні питання до 2 розділу

1. Атрибути-прикмети: їх характеристики.
2. Атрибути-основи: їх характеристики.
3. Що таке поле бази даних та його характеристики?
4. Що таке записи в базі даних, як вони створюються?
5. Яке призначення ключів у базах даних?
6. Які є ключі у базах даних?
7. Які відомі способи створення баз даних?

3. Робота з СУБД Microsoft Access

3.1. Запуск Microsoft Access

Запуск Microsoft Access здійснюється із меню **Пуск / Все програми / Microsoft Office / Microsoft Office Access 2007**.

Виклик програми веде до відкриття вікна запрошення Microsoft Access, що має стандартне для програм Microsoft Office оформлення (рис. 4). У правій частині вікна на панелі **Открыть последнюю базу данных** треба указати яку відкрити базу даних. Якщо вибрати піктограму на центральній панелі **Нова база даних**, справа відкривається однойменна панель, в якій треба задати ім'я нової бази, що створюється, та натиснути кнопку **Створити (Создать)**. Відкриється вікно з назвою нової пустої бази даних.

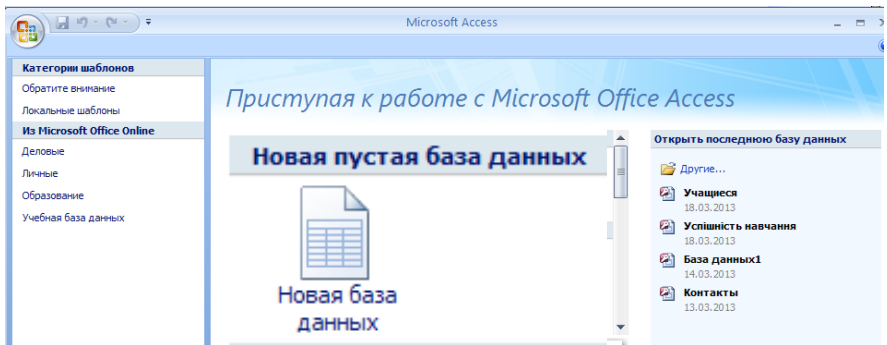


Рис. 4 Вікно запрошення Microsoft Access

Для створення БД можна використовувати шаблони перелічені на центральній панелі вікна Microsoft Access. Категорії шаблонів представлені на панелі у лівої частині вікна.

3.2. Функції СУБД Microsoft Access

Microsoft Access входить до стандартного пакету Microsoft Office і є функціонально повною реляційною СУБД. Це означає, що система повністю виконує всі основні функції СУБД: визначення даних, обробку даних та управління даними. Саме тому Microsoft Access являє собою зручний приклад для конкретного розгляду ідей та методів, покладених в основу побудови СУБД.

До основних можливостей СУБД Microsoft Access можна віднести наступні:

- Проектування базових об'єктів - двовимірні таблиці з полями різних типів даних.
- Створення зв'язків між таблицями, з підтримкою цілісності даних, каскадного оновлення полів і каскадного видалення записів.
- Введення, зберігання, перегляд, сортування, зміна і вибірка даних з таблиць з використанням різних засобів контролю інформації, індексування таблиць і апарату алгебри логіки.
- Створення, модифікація і використання похідних об'єктів (запитань, форм і звітів).
- MS Access дозволяє виконувати ведення бази даних з використанням таких об'єктів: таблиці, запити, форми, звіти, макроси та модулі.

Таблиця – об'єкт, призначений саме для зберігання даних. Кожна таблиця вміщує відомості про суб'єкти певного типу. Поля (стовпці) таблиці призначені для зберігання конкретних характеристик, або

властивостей (атрибутів) суб'єктів. Кожний рядок таблиці складає запис, який містить дані про конкретний суб'єкт (рис. 3). Для кожної таблиці треба визначити первинний ключ, який уніфікує кожний запис. Для таблиці, поданої на рис. 5, використаний первинний ключ – лічильник.

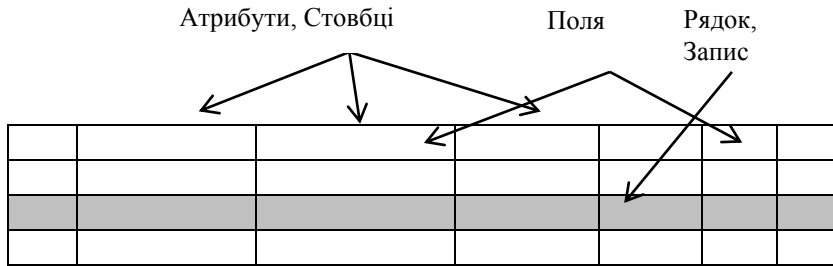


Рис. 5. Приклад таблиці з її елементами

Запит – об'єкт, який дозволяє користувачу одержати потрібні дані з однієї чи декількох таблиць у вигляді, найбільш зручному для сприйняття, виходячи з завдань, що стоять перед користувачем. Залежно від призначення розрізняють запити на вибірку, оновлення, видалення, додавання даних. Запити дозволяють також створювати нові таблиці, використовуючи дані існуючих таблиць.

Форма – об'єкт, призначений, головним чином, для вводу даних, відображення їх на екрані у найбільш зручній формі та керування роботою додатків. Форми також надають можливість користувачу запускати макроси та процедури, написані мовою Visual Basic for Application (VBA), забезпечуючи тим самим додаткову автоматизацію виконання робіт під час ведення баз даних.

Звіт – об'єкт, призначений для форматування, обчислення підсумків, оформлення та друкування вибраних даних.

Макрос – об'єкт, що являє собою структурований опис одної чи декількох дій, які виконуються програмою у відповідь на певну подію. Такий об'єкт зустрічається в усіх інших офісних програмах.

Модуль, Процедура – об'єкт, який містить програми на мові VBA.

Передбачені можливості подання об'єктів у вигляді, який, за думкою користувача, є найбільш зручним та наочним для нього.

Робота з базою починається із створення таблиць.

3.3. Склад інтерфейсу Microsoft Access 2007

Головне вікно програми Microsoft Access складається з наступних елементів:

1. рядок заголовка;
2. рядок меню;
3. панель швидкого доступу
4. кнопка Office
5. панель інструментів;
6. вікно бази даних;
7. область переходів;
8. рядок стану.

1) У рядку заголовка знаходиться системне меню у вигляді піктограмми, розташованій ліворуч від назви головного вікна: «Microsoft Access».

2) Рядок меню містить групи команд об'єднані за функціональною ознакою. Команди, що містять в меню аналогічні командам редактора Word, Excel і в інших програмах Office.

3) Панель швидкого доступу можна переміщувати і налаштовувати шляхом розміщення будь-якої команди з колекції.

4) Кнопка Office дозволяє відкривати, закривати, зберігати БД.

5) Панель інструментів. При запуску Access за замовчуванням виникає одна панель інструментів. На панелі інструментів розташовані найбільш часто використовувані команди. Перед створенням БД необхідно ознайомитися з головним меню та панелі інструментів.

6) Вікно бази даних має:

- робоча область, де розміщуються вкладки відкритих об'єктів;
- вікно властивості об'єктів.

7) Область переходів містить усі об'єкти бази даних.

8) Рядок стану знаходиться внизу головного вікна і призначена для виведення короткої інформації про поточний режим роботи.

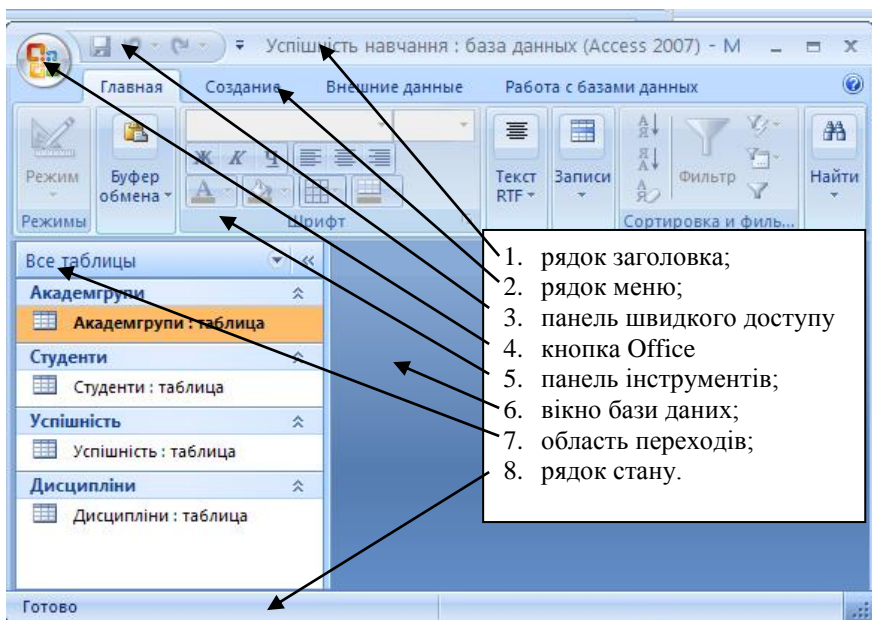


Рис. 6. Головне вікно Microsoft Access 2007 бази даних Успішність навчання.

3.4. Контрольні питання до 3 розділу

1. Коротко охарактеризуйте можливості MS Access 2007.
3. Що таке модуль? Для чого він використовується?
4. Порядок створення бази даних MS Access 2007?
7. Перелік полів та властивостей даних у них.
8. Як створити зв'язки між таблицями?

4. Розробка таблиць у СУБД Microsoft Access

Таблиці є основним об'єктом бази даних. Тому створення бази починається із створення таблиць. Слід нагадати, що перед тим як приступити до створення об'єктів (таблиць, форм, звітів тощо) безпосередньо на ПК, потрібно їх спроектувати, тобто розробити структуру БД. Для цього необхідно попередньо проаналізувати вхідні дані, виділити їх основні складові елементи, тобто спроектувати об'єкт Таблицю: поля таблиці, описати її основні характеристики та параметри. Таким чином створення таблиць БД потребує ретельного їх планування. Таблиці створені без копіткої підготовчої роботи, пізніше, як

правило, потребують зміни її структури, для чого необхідні значні додаткові зусилля і витрати часу.

При цьому треба визначитися у наступному:

- які таблиці повинні входити до складу бази;
- яку структуру повинна мати кожна таблиця;
- які характеристики повинні мати дані у записах;
- який зв'язок між таблицями треба здійснити.

Відповіді на ці питання визначаються під час планування структури бази користувачем. Наступний етап зв'язаний з формуванням бази даних на ПК, тобто безпосередньо роботою з Системою Управління Базами Даних Microsoft Access. Тут зосереджується у структурованому вигляді вся інформація про об'єкти, яка використовується надалі під час роботи СУБД MS Access з конкретним об'єктом.

4.1. Створення таблиці

Створення БД з допомогою СУБД Access починається зі створення структури таблиць і встановлення зв'язків між таблицями. Створення таблиці складається з двох етапів: визначення структури таблиці, тобто визначення потрібних полів та їх властивостей; заповнення таблиці записами. Порядок створення структури таблиці:

- порожня таблиця додається до БД;
- вводяться імена стовпців, типи даних, їх властивості і за потребою опис стовпців;
- встановлюється первинний ключ таблиці;
- створюються індекси для обраних стовпців;
- зберігається структура таблиці;

Створення структури таблиці можливе двома способами: у режимі Таблиці (**Создание / Таблицы / Таблица**) і режимі Конструктор (**Создание / Таблицы / Конструктор таблиц**). Режими легко перемикаються командою **Главная / Режимы / Режим**. У режимі Таблиці структура таблиці створюється зміненням заголовків стовпців подвійним клацанням по ним Миші або натисканням кнопки інструментів **Поля и столбцы / Переименовать** (рис.). У цьому режимі можна задавати тип даних у полі **Форматирование и тип данных / Тип данных (Тип данных)**.

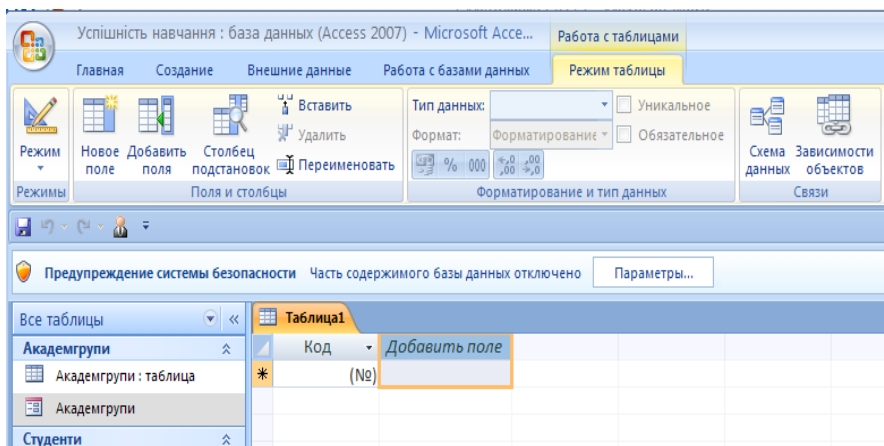


Рис. 7 Режим таблиці.

При виборі режиму Конструктор буде відображено вікно Конструктора таблиць, в якому необхідно ввести список усіх полів і їх типи. Створена таблиця може бути надалі скорегована користувачем у режимі **Конструктор**.

При виборі режиму Конструктор буде відображено вікно Конструктора таблиць, в якому необхідно ввести список усіх полів і їх типи.

У перший рядок колонки **Ім'я поля** вводимо код студентів (Код студента) і натискаємо клавішу Enter, при цьому курсор переміститься в колонку Тип даних, де з розкривного списку вибираємо тип даних - Лічильник. Потім натискаємо клавішу Enter, при цьому курсор переміститься в колонку Опис, при необхідності вводимо опис даних, які будуть вводитися в це поле таблиці.

Визначаємо перший рядок таблиці (поле Код) як поле первинного ключа, для цього клацніть на піктограмі **Конструктор/Сервіс/Ключове поле**. Зліва від імені поля з'явиться зображення ключа. Якщо поле є ключовим, тобто полем первинного ключа, то властивості Індексоване поле присвоюється значення Так (збіги не допускаються).

Далі у другий рядок Ім'я поля вводимо код групи (Код Групи) і вибираємо тип даних - числовий. Призначаємо це поле полем Зовнішнього ключа, для цього необхідно виділити поле Код Групи і в області властивостей цього поля в рядку Індексоване поле зі списку вибрати значення Так (Збіг допускається).

Потім у третій, четвертий і п'ятий рядок Ім'я поля вводимо Ім'я, Прізвище, По батькові і вибираємо тип даних текстовий. При цьому в нижній частині екрана в розділі Властивості поля з'являється інформація про властивості даного поля. При необхідності туди можна вносити зміни, виконавши клацніть у відповідному рядку, видаливши попереднє значення і ввівши нове. Найбільш часто використовуються властивості Розмір поля і Обов'язкове поле. Перше впливає на розмір БД, а друге вказує чи обов'язково вводити дані у це поле.

Далі створюються інші поля згідно з даними у моделі "сутність зв'язок". Для числових полів важливими є властивості **Розмір поля, Маска ввoda, Умови на значення, Обов'язкове поле**.

Після створення структури таблиці необхідно зберегти її. Натиснути кнопку Office і вибрати команду Зберегти або Зберегти як... У вікні Збереження ввести ім'я для створення таблиці: Студенти, ОК.

Далі створюються структури інших таблиць: Академгрупи, Дисципліни, Успішність.

Після створення структури таблиць необхідно встановити зв'язок між ними.

При виборі режиму Конструктор відкривається вікно Таблиця 1: таблиця. У верхній частині вікна послідовно у запропонованих рядках записуються імена полів спланованої таблиці, вибирається тип даних та, за необхідності, у стовпці **Описание** вводиться пояснювальний текст для кожного поля (рис. 1.1).

Імена стовпців повинні бути інформативними, правильно відображати їх призначення і не бути особливо довгими. Слід так добирати імена, щоб не доводилося їх змінювати, тому що це може призвести до додаткових проблем, а інколи й до неправильної роботи БД. В системі Access 2007 можна використовувати в іменах букви, цифри і спеціальні символи. У той же час використання деяких із них заборонено. Тому для запобігання помилок краще в іменах таблиць застосовувати тільки букви і цифри.

Тип для кожного поля вибирається в стовпці Тип даних із списку, що розкривається у кожному рядку кнопкою (рис. 1.2). За замовчанням програма задає тип **Текстовий**. Наведемо деякі правила та рекомендації щодо роботи з типами даних:

— дані, що вводяться в певну клітинку стовпця таблиці, повинні відповідати оголошеному типу стовпця. Наприклад, якщо стовпцю присвоєно тип текст, а в будь-яку його клітинку робиться спроба ввести число, то буде видано повідомлення про помилку введення;

— не слід зловживати використанням типу Примітка. Для збереження даних цього типу потрібні значні обсяги пам'яті, тому слід використовувати його тільки у разі невідкладної потреби. Слід пам'ятати, що сортувати й індексувати записи за цими полями неможливо;

— не слід зберігати цілі невеликі числа у стовпцях довгого типу, інакше це призведе до додаткового використання пам'яті;

— не слід використовувати тип текст для зберігання даних типу Дата/час, доцільно використовувати саме цей тип;

— у деяких випадках виникає необхідність подання числових даних як текстових і навпаки. Для цього перед сортуванням необхідно використати функцію перетворення типу даних;

— слід визначити типи стовпців так, щоб потім їх не змінювати. Якщо, все ж таки, тип даних у стовпці змінюється, то дані, які зберігаються в ньому, будуть перетворюватися автоматично. Але слід враховувати, що в багатьох випадках можуть при цьому виникати помилки. Наприклад, у ході перетворення даних стовпця грошового типу в числовий тип Довге ціле частина даних втрачається, тому що відкидається дробова частина. Перетворення даних типу OLE у будь-який тип не допустиме.

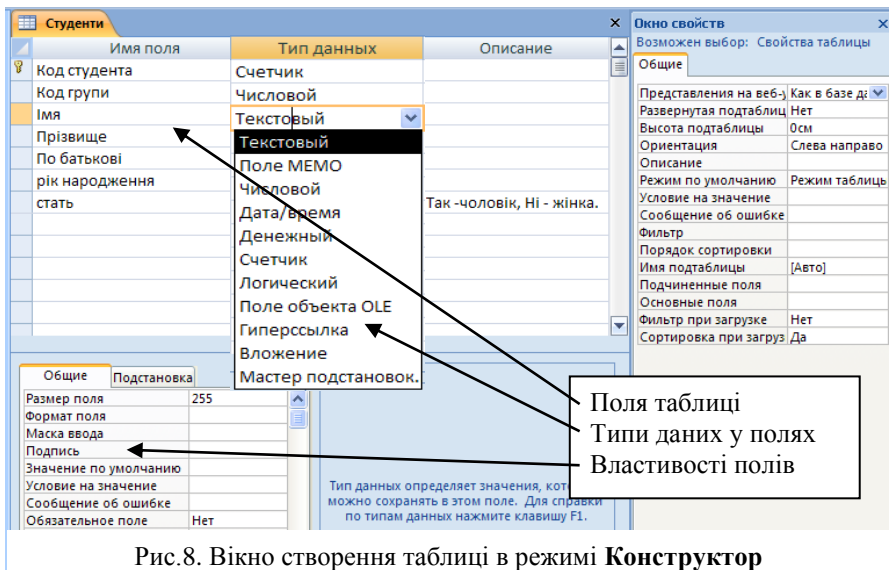


Рис.8. Вікно створення таблиці в режимі **Конструктор**

Тип даних полів визначає вигляд та діапазон допустимих значень даних кожного поля, а також обсяг пам'яті, який виділяється для ньо-

го. Перелік передбачених типів даних полів наведений в табл. 1. Для кожного поля відповідно до типу даного в ньому можна задати певні властивості поля на панелі **Свойства поля**, що розташовано у нижній частині вікна. Перелік властивостей відповідає заданому типу даних.

Таблиця 1 Типи даних.

Тип даних	Коротка характеристика
Текстовий	Текст та числа (може містити до 255 символів). Застосовується в адресах, прізвищах, описах тощо. Його рекомендується застосовувати також для запису даних, які містять тільки цифри, над якими не виконуються арифметичні операції.
Поле Мемо	Довгий текст та числа (коментарі, пояснення тощо). Може містити до 65536 символів
Числовий	Загальний тип для числових даних. Допускає проведення математичних розрахунків та встановлення різних типів числових даних в полі Свойства (клітина Размер поля)
Дата/час	Значення дати та часу. (до 8 байтів)
Денежний	Грошові значення. Містить 15 десяткових цифр ліворуч і 4 цифри праворуч від крапки. Значення цього типу не округлюється.
Лічильник (Счетчик)	Зберігаються цілі числа довжиною до 4-х байтів. При додаванні нового рядка в таблицю автоматично у це поле записується число.
Логічний	Це логічний тип, за яким зберігається в пам'яті 1 або 0, що еквівалентно значенням істинно (так) або хибно (ні). "True"/"False", "Вкл"/"Викл".
Поле об'єкта OLE	У полях цього типу можуть зберігатися документи Word, робочі аркуші Excel, зображення, аудіо- і відеокліпи довжиною до 1 ГБ. Поля OLE не індексуються, не можна за цим типом виконувати й сортування.
Гіперпосилання	Містить адресу джерела в мережі Інтернет або на жорсткому диску. Поле містить до 64000 символів, яке може складатися з трьох частин: текст, який, зазвичай, виводиться підкресленим; адреса URL або URN файлу в Інтернеті або на жорсткому диску; внутрішня адреса, яка вказує на точку у файлі або на сторінку, наприклад, ім'я форми або звіту Access.

Значення властивостей полів можна вибрати з наведеного переліку (перелік з'являється при виборі відповідного рядку таблиці **Властивості поля (Свойства поля)** або задавати за допомогою вікна **Построитель выражений**, активізувавши його кнопкою (...) у відповідному рядку.

Властивості залежать від типу даних. Можливі значення властивостей наведені в табл. 2. Типи числових даних наведені у табл. 3.

Для полів **дата/час** в області **Властивості поля** можуть бути встановлені такі формати цього типу:

- повний формат дати: 23.04.2012 19:24:37;
- довгий формат дати 20 серпня 2009;
- середній формат дати 15-вер-09;
- короткий формат дати 28.05.2015;
- довгий формат часу: 20:35:29;
- середній формат часу: 6:05;
- короткий формат часу: 15:31.

У полі лічильник спосіб запису числа такий: якщо в області **Властивості поля** властивості **Нові** значення встановлено значення **Поступово**, то значення цього поля буде збільшене на одиницю відносно попереднього запису. Якщо ж встановлено **Випадково**, то буде записане випадкове ціле число.

Набір властивостей може встановлюватися також для таблиці в цілому. На відміну від властивостей полів, для таблиці визначаються її властивості в цілому, а саме її вигляд та поведінку. Для завдання конкретних значень властивостей таблиці її необхідно відкрити в режимі **Конструктор** та вибрати команду **Конструктор / Показати или скрыть / Страница свойств**. В бланку властивостей, що з'явиться при цьому (**Окно свойств**), задаються значення властивостей таблиці.

Таблиця 2 Властивості поля.

Властивості	Можливі значення властивостей
Розмір поля (Field Size)	Максимальна довжина тексту або тип подання чисел
Формат поля (Format)	Формат подання даних (використовується один із стандартних або створюється спеціальний)
Дробна частина (Decimal Places)	Число знаків справа від коми
Маска вводу (Input Mask)	Символи форматування для введення даних (використовуються стандартні або створюються)

	спеціальні)
Підпис (Caption)	Стандартний підпис поля, що відтворюється в формі або звіті
Значення по умовчанию (Default Value)	Значення, що автоматично вводиться в поле при створенні запису
Умови на значення (Validation Rule)	Вираз, що визначає умову для введення даних
Повідомлення про помилку (Validation Text)	Текст повідомлення, який виводиться при введенні даних в поле з порушенням умови на їх значення
Обов'язкове поле (Required)	Значення, що визначає можливість введення пустих значень

Таблиця 3. Розмір даних типу Число

Тип	Діапазон чисел	Потрібна пам'ять
Байт	0 - 255	1 байт
Ціле число	-32768 - 32767	2 байти
Довге ціле число	-2147483648 - 2147483647	4 байти
Одинарне значення	-3.4E38 - 3,4E38	4 байти
Подвійне значення	-1.797E308 - 1Д97E308	8 байтів
Ідентифікатор реплікації		16 байтів
Десятковий		8 байтів

При створенні таблиці слід мати на увазі, що дуже важливим є забезпечення в ній достовірності даних. Треба передбачити захист таблиці від занесення хибних даних.

Певний захист від грубих помилок забезпечується вже тим, що для кожного даного визначається його тип. Спроба занести у поле таблиці дані іншого типу (наприклад, текстового замість числового або дати) викличе реакцію програми з попередженням про помилку.

Для перевірки достовірності форматованих даних можна використовувати маски вводу. Вони є своєрідними підказками і призначені для полегшення введення даних, оскільки задають формат запису даних. Невідповідність даних, що вводяться, завданій масці веде до по-

передження з боку програми і тим самим виключає помилки з боку користувача.

Для створення маски необхідно в режимі Конструктор вибрати поле, для якого створюється маска, встановити курсор в рядок **Маска вводу** (панель **Властивості поля**) та сконструювати маску, користуючись символами табл. 4.

Користувач, за бажанням, може корегувати запропоновані маски або вибирати їх без змін. Він також може створити маску самостійно, записуючи у відповідному рядку набір символів. Деякі використовувані при завданні маски символи наведені у табл. 1.4.

Таблиця 4

Символ маски	Описання
9	В позицію може бути введена цифра або пропуск
0	В позицію обов'язково повинна бути введена цифра
L	В позицію має бути введена довільна літера
#	В позицію може бути введена цифра, пропуск, знак + або -
?	В позицію може бути введена довільна літера
A	В позицію має бути введена літера або цифра
&	В позицію має бути введений довільний символ або пропуск
>	Перетворює всі символи праворуч до верхнього регістру
<	Те ж до нижнього регістру

Прикладами масок можуть бути: 00/00/00 (для запису дати народження), 99LLA (для введення тексту, що починається з цифр), 000-00-00 (для запису номера телефону).

Ще одним способом захисту таблиці від введення хибних даних є запис у властивостях полів умови на значення (рядок властивостей **Умови на значення**). Користувач може сконструювати умови, причому досить складні, перевірки даних при введенні їх у таблицю. Можна також передбачити повідомлення з боку машини, якщо дані не відповідають умові (**Повідомлення про помилку**). Приклади запису умов на значення наведені у табл. 5.

Нагадаємо, що символ * позначає будь-яку кількість довільних символів, символ ? – один довільний символ, символ # – одну довільну цифру.

З наведених прикладів випливає, що умови на значення конструюються за допомогою знаків порівнянь, логічних функцій та стандартних позначень символів.

Таблиця 5

Умова на значення	Описання
≥ 1000	Дане, що вводиться, має бути не менше завданого числа
between 34 and 123	Дане має лежати у діапазоні від 34 до 123
5 or 9	Може бути занесене одне з двох вказаних чисел
not 1937	Заборонено запис завданого числа
Like [A-Г]*	Назва починається з літер А, Б, В, Г
not Іван	Заборонена вказана назва
Пет???	Запис починається з Пет та має ще три літери
$\leq 24-10-37$	Дата не може бути ранішньою за завдану
Len([назва поля]) = 6	Дане у полі повинне мати шість символів

Ще одним важливим моментом створення таблиці є визначення її поля первинного ключа (ключового поля). Для обґрунтованого вибору ключового поля доцільно звернутися до чернетки бази даних, що має бути розробленою заздалегідь. Тут обов'язково має бути визначений зв'язок таблиць між собою та ключові поля у таблицях, які будуть цей зв'язок забезпечувати.

Після визначення всіх полів таблиці у режимі Конструктор її треба зберегти. При цьому, якщо ключове поле не було задане, програма попередить про це і запропонує свої послуги: створити ключ у вигляді *Счетчик*. Програма запропонує також надати ім'я таблиці. Доцільно надати змістовне ім'я, що відбиває суть таблиці.

У списку типів даних останній рядок це **Майстер підстановок**. Це не тип даних. За допомогою майстра робиться поле доступним для підстановок. Це визначає, що будь-яке поле цього стовпця може набувати тільки одного з допустимих значень. За допомогою Майстра можна задати фіксований набір допустимих значень або взяти його з другої таблиці або запиту. Вікно майстра **Создание подстановки зображено на рис.**

Таку ж підстановку можна зробити на вкладці **Підстановка** панелі **Властивості поля**. Тип елемента управління слід вибрати **Список** або **Поле зі списком**. Коли **Тип джерела (Тип источника)** має

значення **Таблиця або запит** то підстановка здійснюється з таблиці або запиту, ім'я якого вказано у наступному рядку **Джерело строк**. Якщо **Тип джерела** має значення **Список значень**, то підстановка здійснюється зі списку у наступному рядку **Джерело строк**.

Підстановку можна створити також при створенні таблиці методом **Таблиця**. Для цього командою **Поля и столбцы / Столбец подстановок** відкривається вікно **Создание подстановки** (рис. 9).

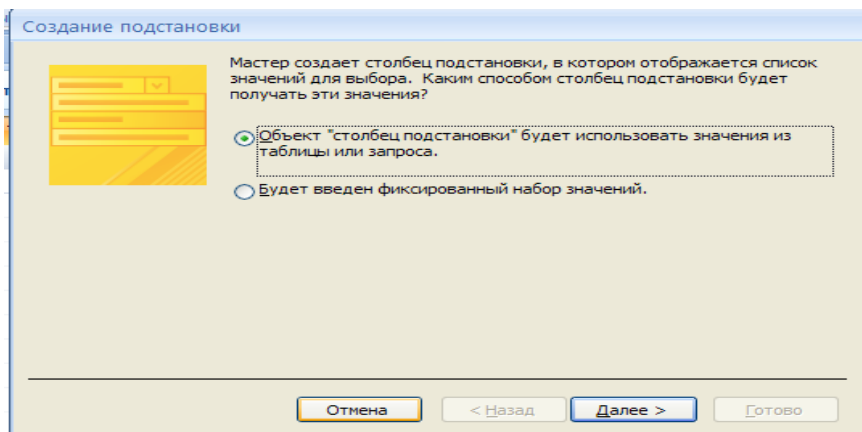


Рис. 9 Вікно майстра Створення підстановки.

У цьому вікні треба вибрати звідки брати список для підстановки. Далі слід вказати список або таблицю і поля як джерело підстановки.

Для створення таблиць можна використовувати шаблони, які мають типовий набір полів. Кнопка **Создание / Таблицы / Шаблоны таблиц**.

4.2. Ключові поля та індекси

Нагадаємо, що кожна таблиця повинна мати ключове поле, тобто поле, значення якого не повторюється ні в якому іншому запису. Таблиця може містити кілька ключових полів, але використовується тільки одне з них. Таке поле називають **первинним ключем**. Найвність у кожному запису унікального ключа називають сутнісною цільністю. Головна вимога до первинного ключа — унікальність його значень. У більшості випадків первинний ключ складається з одного стовпця. Найчастіше в роль первинного ключа використову-

ється поле типу **Лічильник**. Однак інколи розробники відмовляються від стовпця типу **Лічильник** і намагаються створити первинний ключ із природних стовпців таблиці, тобто стовпців, які вже є в таблиці. Якщо в таблиці немає жодного унікального стовпця, для первинного ключа обирається два або більше стовпців. Такі первинні ключі називають складеними.

Наприклад, у таблиці, що містить прізвища студентів, поле **Прізвище** не можна визначити як первинний ключ, тому що в університеті можуть бути студенти з однаковими прізвищами. А, наприклад, поля **Прізвище** й **Ім'я** співпадають значно менше і можуть бути складним ключем. Але при великій кількості студентів вірогідність спів падання збільшується і потрібно ускладнювати ключ.

Зазначимо, що складний первинний ключ у реальних задачах застосовується рідко, тому що ускладнюється сама база даних, а також ускладнюється підтримка зв'язків між таблицями.

Для створення первинного ключа необхідно відкрити таблицю в режимі конструктора, виділити поле, яке використовується як первинний ключ, і натиснути кнопку **Ключове поле**, яка знаходиться в розділі **Сервіс** вкладки **Конструктор**. Крім того, створити первинний ключ можна й за допомогою команди **Ключове поле** контекстного меню певного стовпця.

Для створення первинного ключа типу **Лічильник** у відкритій таблиці в режимі конструктора необхідно додати до таблиці стовпець, надати йому тип **Лічильник**, відмітити його й натиснути на кнопку **Ключове поле**. Крім того, первинний ключ для поля цього типу може бути встановлений і в тому разі, якщо під час створення таблиці взагалі ключ не визначено.

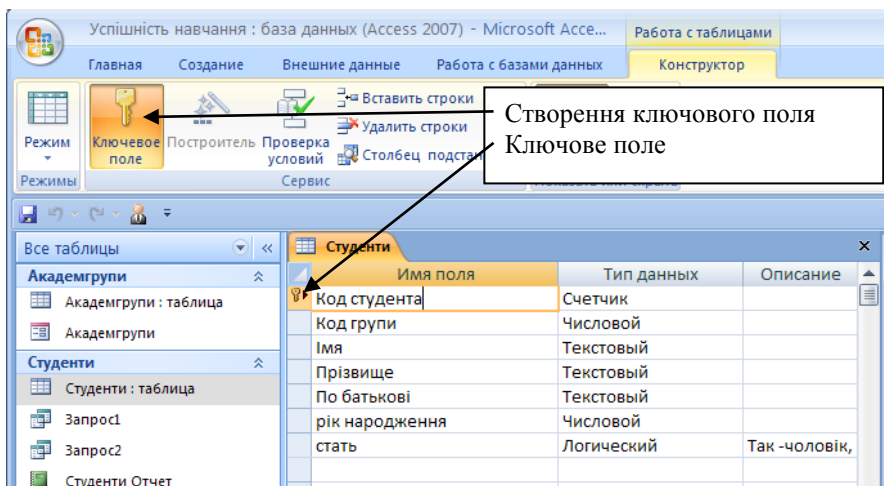


Рис. 10 Ключовое поле таблиці.

Перейдемо тепер до розгляду індексування таблиць. Дані у таблиці зберігаються упорядкованими за значенням первинного ключа. Нагадаємо, що індексування — це процес створення додаткових таблиць для певного поля. Ці таблиці, як правило, містять тільки один стовпець, у якому зберігаються вказівники на певні рядки таблиці. За допомогою вказівників визначають порядок розміщення рядків, упорядкованих за значенням цього поля.

Отже, індексна таблиця — це додаткова таблиця, де зберігається не фізичний, а логічний порядок записів за значенням конкретного стовпця. Таких індексних таблиць для таблиці може бути кілька, наприклад, за прізвищами, за кількістю учнів тощо. Найчастіше використовуються прості індексні таблиці, які складаються з одного стовпця.

Головне призначення індексних таблиць — підвищення швидкості пошуку необхідних даних (інколи вона підвищується до 5 разів). Щоб знайти деякий запис, в Access спочатку знаходиться його положення в індексі, потім вибирається із нього місце запису в таблиці, які використовуються для пошуку даних. Не слід індексувати стовпці, значення яких часто змінюються. Слід також пам'ятати, що під час додавання до таблиці нового запису або в разі вилучення запису, оновлюються ці індексні таблиці, що призводить до втрати продуктивності бази даних. Бажано для таблиці мати не більше 5-6 індексних таблиць.

Для створення простої індексної таблиці певного поля необхідно надати його атрибутіві **Індексоване поле** одне із значень:

- так (допускається співпадання);
- так (співпадання не допускаються).

Зазначимо, що первинний ключ в Access 2007 завжди індексований. За замовчуванням записи таблиці виводяться відсортованими за значенням цього ключа. У процесі введення даних у таблицю обов'язково перевіряється значення первинного ключа на дублювання. Якщо значення дублюється, введення запису блокується. Значення первинного ключа типу Лічильник у процесі введення даних формується автоматично.

4.3. Установка зв'язків між таблицями в СУБД Access

Після створення структури таблиць для сутностей бази даних необхідно встановити зв'язки між таблицями. Зв'язку між таблицями в БД використовуються при формуванні запитів, розробки форм, при створенні звітів.

Реляційна база даних складається з багатьох взаємопов'язаних таблиць. При зв'язуванні двох таблиць в одній (головній) первинний ключ зв'язується із зовнішнім ключем другої (підлеглої) таблиці. Зовнішнім ключем називають поле, значення якого співпадають із значеннями первинного ключа головної таблиці. Імена первинного ключа і зовнішнього ключа можуть бути різними. Головне, щоб їхні значення співпадали.

Часто первинний ключ першої таблиці штучно вводять в другу таблицю саме з метою їх зв'язування. Але цей ключ не є первинним у другій таблиці, тому що його значення тут можуть повторюватися. Тому його й називають зовнішнім ключем. Наприклад, поле **Код групи**, яке є первинним ключем таблиці **Академгрупи** і зовнішнім ключем таблиці **Студенти**. Його значення в останній таблиці дублюється, тому що в одній групі знаходиться багато студентів. Між цими таблицями існує зв'язок типу один — до багатьох (1-∞).

Для створення зв'язків необхідно закрити всі таблиці і вибрати команду **Схема даних** з меню **Робота с базами даних**, з'явиться діалогове вікно **Додавання таблиці** на тлі неактивного вікна **Схема даних**. Це вікно також відкривається командою контекстного меню **Додати таблиці**.

У діалоговому вікні **Додавання таблиць** необхідно виділити імена таблиць і натиснути кнопку **Додати**, при цьому у вікні **Схема даних** додаються таблиці. Після появи всіх таблиць у вікні **Схема да-**

них необхідно закрити вікно **Додавання таблиці**, клацанням лівою кнопкою миші на кнопку **Закрити**.

Наступний крок - це встановлення зв'язків між таблицями у вікні. Для цього у вікні **Схема даних** необхідно відбуксувати (пересунути) поле **Код Групи** з таблиці **Академгрупи** на відповідне поле таблиці **Студенти**, в результаті цієї операції з'явиться вікно **Зміна зв'язків** (рис. 11).

У вікні діалогу **Зміна зв'язків** необхідно активізувати прапорці: **Забезпечити цілісність даних, каскадне оновлення пов'язаних полів і каскадне видалення пов'язаних записів**, переконатися в тому, що встановлено тип відносин **один-до-багатьом** і натиснути кнопку **Створити (Создать)**.

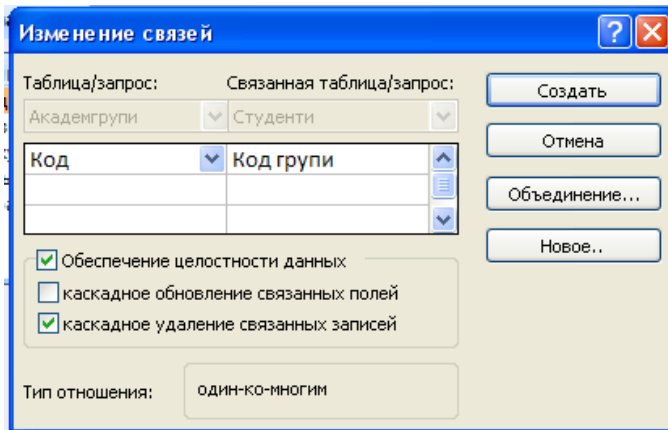


Рис. 11. Вікно **Зміна зв'язків**

У цьому вікні встановлюються прапорці **Забезпечення цілісності даних**, а потім — **Каскадне видалення пов'язаних записів**. Якщо останній прапорець буде знято, то неможливо буде вилучити записи із головної таблиці до тих пір, поки вручну не буде знятий зв'язок між усіма пов'язаними з нею записами.

Якщо прапорець **Забезпечення цілісності даних** не встановлено, можна в таблиці додавати нові записи, змінювати ключові поля і вилучати пов'язані записи без попередження про порушення цілісності. У нормальному режимі роботи з базою даних цілісність повинна бути встановлена. Коли цей прапорець встановлено, стають доступні два прапорці: **Каскадне оновлення пов'язаних полів і Каскадне видалення пов'язаних полів**.

Якщо встановлено перший прапорець, то в разі зміни значення одного з пов'язаних полів у батьківській таблиці нове значення буде автоматично встановлене на весь ланцюжок пов'язаних таблиць. Якщо цей прапорець знятий, змінити поле первинного ключа батьківської таблиці, коли з ним пов'язаний хоча б один запис дочірньої таблиці неможливо. Якщо батьківська таблиця пов'язана з кількома дочірніми, то цей прапорець повинен бути встановлений для кожної таблиці, інакше каскадне оновлення буде заборонено.

Якщо встановлено прапорець **Каскадне видалення пов'язаних полів**, то в разі вилучення батьківського запису автоматично будуть вилучені всі пов'язані дочірні записи. Тому користуватися цим режимом треба обережно. Щоб застосувати режим каскадного вилучення, необхідно встановити прапорець **Каскадне видалення пов'язаних полів** для всіх таблиць. Інакше не буде виконуватися каскадне вилучення ні в одній таблиці.

У вікні **Схема даних** з'явиться зв'язок один-до-багатьом між таблицями **Акадегрупи** і **Студенти**. Аналогічним чином треба зв'язати поля **Код Студента** в таблицях **Студенти** і **Успішність**, а потім поля **Код Дисципліни** в таблицях **Успішність** і **Дисципліни**. У результаті отримаємо **Схему даних**, представлену на рис. 12.

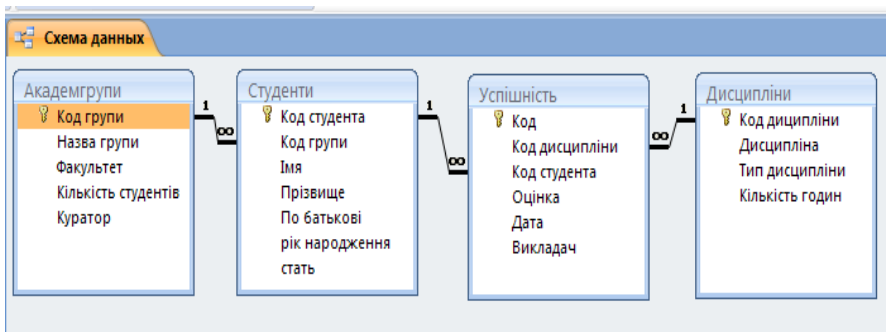


Рис. 12 Таблиці та зв'язки бази даних.

Після встановлення зв'язків між таблицями, вікно **Схема даних** необхідно закрити. Далі необхідно здійснити заповнення всіх таблиць. Заповнення таблиць доцільно починати з таблиці **Академгрупи**, так як полі **Код групи** таблиці **Студенти** використовується як стовпчика підстановки для заповнення відповідного поля таблиці **Студенти**.

Для вилучення зв'язку між таблицями необхідно встановити курсор на лінії зв'язку, натиснути кнопку миші, а потім - на клавішу Del. Можна також відкрити контекстне меню лінії зв'язку (клацнути пра-

вою кнопкою миши по лінії зв'язку) і виконати команду **Видалити (Удалить)**. Для змінення зв'язків треба у контекстному меню лінії зв'язку вибрати команду **Изменить связь**.

Повернемося ще раз до вікна **Змінення зв'язків (Изменение связей)**. Далі слід натиснути у цьому вікні кнопку **Об'єднання (Объединение)**. З'явиться вікно **Параметри об'єднання** (рис. 13). За замовчуванням встановлюється перший тип об'єднання, яке називають об'єднанням за еквівалентністю. Як правило, розробники баз даних встановлюють відношення за еквівалентністю. Потім потрібно натиснути у цьому вікні кнопку ОК і закрити вікно.

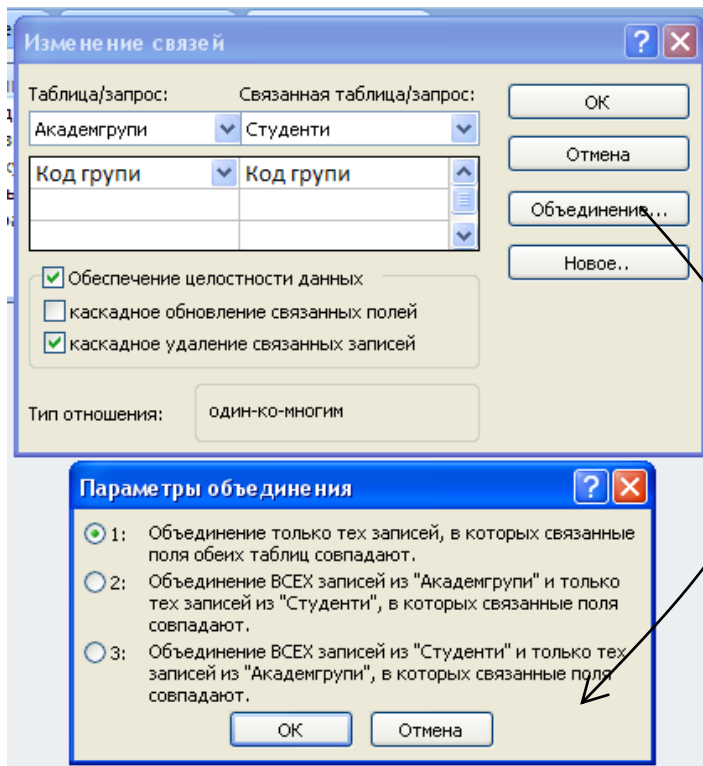


Рис. 13 . Вікно **Параметры объединения**.

4.4. Коректування таблиці

Модифікувати структуру таблиць краще виконувати до внесення в таблиці даних. Коректування створеної таблиці передбачає виконання таких операцій над її структурою: введення нових полів, зміна

їх імен, переміщення та видалення полів, зміна типу даних та їх значень, змінювати імена стовпців; змінювати розміри полів; перетворювати типи даних.

Слід також пам'ятати, що в пов'язаних таблицях стовпці, які є первинним або зовнішнім ключем, вилучити неможливо до тих пір, поки не буде вилучено зв'язок між таблицями.

При корегуванні таблиці слід бути уважним та чітко уявляти функціональну суть змін, що вносяться. При виконанні деяких операцій можуть порушуватися встановлені зв'язки за раніше визначеними ознаками і структурна цілісність таблиці, що може призвести до неповоротної втрати даних та порушення функцій управління базою даних в цілому.

Введення нових полів в таблицю здійснюється в режимі Конструктор. Курсор необхідно встановити в перший пустий рядок в кінці таблиці, вказати назву поля, тип даних та задати властивості поля (за необхідності визначається описання поля). Нове поле можна ввести і в будь-яке місце серед існуючих полів. Для цього необхідно встановити курсор у рядку поля, перед яким додається нове, та вибрати команду контекстного меню (правий клавіш миші) **Добавить строки**. В таблицю в обраному місці вставляється пустий рядок. Таким же чином видаляється зайве поле (**Удалить строки**).

Зміна імен полів здійснюється також в режимі Конструктор шляхом зазначення поля, ім'я якого необхідно змінити, та безпосереднього корегування самого імені. При цьому слід обов'язково оновлювати всі посилання на це поле, якщо вони існують, у всіх інших об'єктах бази даних.

Переміщення полів в таблиці здійснюється для зміни порядку виводу стовпців таблиці в режимі **Таблица**. Для виконання цієї операції слід виділити те поле, яке необхідно перемістити, встановити курсор миші в межах виділення рядків, натиснути і не відпускати лівий клавіш миші. Над останнім виділеним рядком виводиться тонка горизонтальна лінія, яку необхідно за допомогою миші перетягнути під те поле, під яке треба вставити виділене поле. Порядок рядків таблиці змінюється. При цьому змін полів таблиці у режимі Конструктор не відбувається.

При видаленні поля таблиці будуть втрачені всі дані, які воно містить. З видаленням полів таблиці необхідно видаляти всі посилання на нього у всіх інших об'єктах бази даних.

Зміна типу даних вимагає підвищеної уваги. Здійснюється в режимі Конструктор відкриттям списку типу даних у стовпці **Тип дан-**

них та вибором нового типу. Для збереження внесених змін слід виконати команду **Сохранить**. При цьому виконується операція перетворення всіх даних, які містяться в обраному полі, відповідно до нового типу.

Якщо поле, для якого виконується вказана операція, містить дані, то це може призвести до внесення суттєвих помилок або до втрати даних.

Користувач за своїм бажанням може змінити зовнішній вигляд таблиці: розміри рядків та стовпців, кольорове оформлення елементів таблиці, шрифти записів.

Зміна розмірів стовпців та рядків найпростіше виконується за допомогою курсору, фіксуючи його лівим клавішем миші.

4.5. Операції з таблицями

В системі Access 2007 можна виконувати такі операції над таблицями:

- перейменування;
- вилучення;
- копіювання.

Копіювати таблицю можна як у поточну базу даних, так і в іншу базу даних. Розглянемо сутність перерахованих операцій.

Змінити ім'я таблиці можна за допомогою її контекстного меню. У контекстному виконується команда **Перейменувати**. У текстове поле цієї таблиці необхідно ввести нове ім'я й зберегти таблицю. Слід пам'ятати, що після цього потрібно змінити це ім'я в усіх об'єктах бази даних (запитах, звітах, формах і макросах).

Для вилучення таблиці можна скористатися її контекстним меню, у якому необхідно виконати команду **Видалити**.

Один із способів отримання копій таблиці такий:

— викликається контекстне меню таблиці (нагадаємо, що для цього в області переходів встановлюється курсор на імені цієї таблиці й натискається кнопка миші);

— у контекстному меню виконується команда Копіювати;

— викликається ще раз контекстне меню цієї таблиці й виконується команда Вставити.

— у поле Ім'я таблиці цього вікна слід ввести ім'я нової таблиці;

Одним із перемикачів встановлюється режим вставляння:

— Лише структура. У цьому випадку створюється нова порожня таблиця, яка має ту саму структуру, що й таблиця, що копіюється;

—Структура та дані. Створюється повна копія структури таблиці і всіх даних, які зберігаються в ній;

—Додавання даних до наявної таблиці. У цьому випадку записи таблиці, що копіюються, додаються в кінець існуючої таблиці такої самої структури.

4.6. Контрольні питання до 4 розділу

1. Які дії необхідно виконати для встановлення в таблицю нового стовпця?
2. Яким вимогам повинен відповідати первинний ключ?
3. Які переваги ключа типу Лічильник?
4. Як створити первинний ключ у системі Ассезз?
5. Які є способи пошуку записів у таблицях?
6. Які операції можна виконувати зі знайденим записом?
7. Як можна приховати стовпці на екрані?
8. Як можна вилучити знайдені записи?
9. За якими полями зв'язуються таблиці?
10. Як можна ліквідувати і відновити зв'язки між таблицями?
11. Що таке забезпечення цілісності даних?
12. Як можна переглянути всі зв'язки між таблицями?
13. У якому режимі вводяться дані з клавіатури в таблицю?

5. Робота з даними у таблицях

5.1. Відкривання таблиці та навігація

Для відкривання таблиці указати в області переходів потрібну таблицю і виконати команду контекстного меню **Відкрити (Открыть)**. Можна також відкрити двійним клацанням миши. На екрані з'явиться структура таблиці БД в режимі Таблиці. Нова таблиця складається з одного порожнього рядку.

Після вводу даних у поле натисканням клавіші Enter переміщуємо курсор на інше поле. Після вводу першої запису порожній запис зміщується в кінець таблиці. Перехід до наступного поля (зліва направо) здійснюється також натисненням клавіші Tab, а у зворотному напрямку Shift+Tab.

У програмі Access застосовуються різні методи переміщення по таблиці в режимі таблиці. Переходити від запису до запису можна за допомогою: клавіш управління курсором або навігаційних кнопок унизу вікна таблиці (рис. 14).

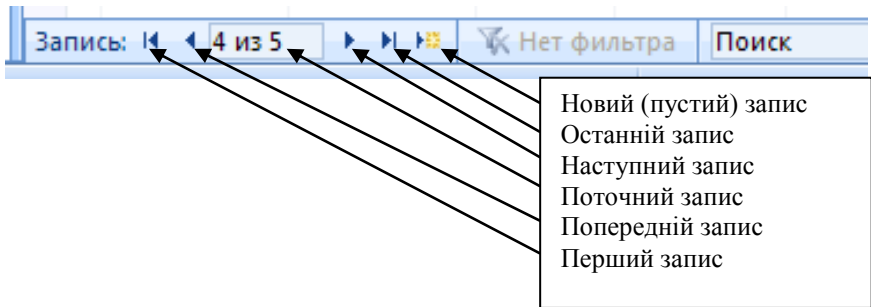


Рис. 14 Навігаційна панель таблиці.

Кнопки навігації застосовуються для переміщення курсору в перший запис, попередній, наступний або останній запис. Навігацію в таблиці можна також здійснювати за допомогою миші, смуг прокручування і комбінації деяких клавіш.

5.2. Введення, редагування і видалення записів

В таблицю дані вводяться шляхом додавання нових записів у режимі таблиці. В пустий рядок послідовно у всі поля вводяться дані. Запис автоматично зберігається при переході на наступний рядок або при закриванні таблиці. Перехід на новий пустий рядок здійснюється навігаційною кнопкою **Новая запись**. Слід пам'ятати, що вводити у стовпці можна тільки ті типи даних, які співпадають з визначеним типом стовпця. Поле лічильника заповнюється автоматично.

Для зменшення кількості помилок введення даних використовуються властивості поля (рис. 15), які встановлюються у режимі Конструктор.

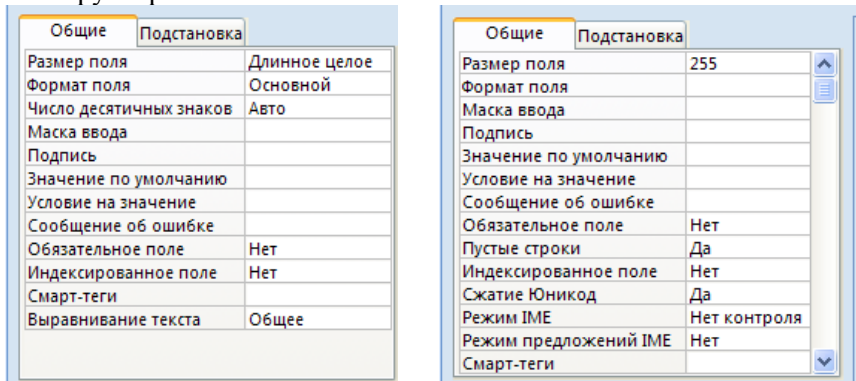


Рис. 15 Властивості числового і текстового поля.


З метою прискорення створення записів є можливість введення в поле значення за допомогою властивості **Значение по умолчанию**. Таке значення автоматично вводиться в поле при створенні кожного нового запису. Це не впливає на вже введені в поле дані та може бути замінено будь-яким іншим значенням шляхом ручного введення. Для визначення такого способу введення даних необхідно в режимі **Конструктор таблицы** для цього поля в рядку властивості **Значение по умолчанию** вказати, яке саме значення буде вводиться за замовчуванням.

Якщо на значення даних в полі введені обмеження у рядку властивостей **Умови на значення (Условие на значение)** і при введенні даних не були виконані ці умови, то буде видаватися текст, що записаний в рядку властивостей **Сообщение об ошибке**.

При введенні даних значення полів можна залишати пустими. Така необхідність часто виникає у зв'язку з відсутністю даних або їх «нульовими» значеннями на момент створення бази даних. Для цього у властивостях поля **Обязательные данные** вказується «Ні». Надалі при коректуванні таблиць пусті поля можна заповнювати даними. У режимі таблиці можна змінювати деякі параметри виведення даних, наприклад, розмір шрифту, ширину стовпців, висоту рядків.

В СУБД Access передбачена також можливість введення даних шляхом копіювання запису з наступним його вставленням через команду **Вставить**.

В процесі введення даних автоматично перевіряються такі типи даних: числові і грошові, дати-часу і логічні. Крім того, можна задавати додаткові правила перевірки за допомогою атрибутів **Формат поля** (для числових даних), **Маска введення** (для полів дата/час та текстових) і **Умови на значення**. Перевірка здійснюється в момент переходу до нового поля. Наприклад, якщо в поле типу **Число** введені букви, то у разі виходу із цього поля з'явиться повідомлення.

Умови створюються за допомогою майстра **Построитель выражений**, який активізується кнопкою  у рядку властивостей.

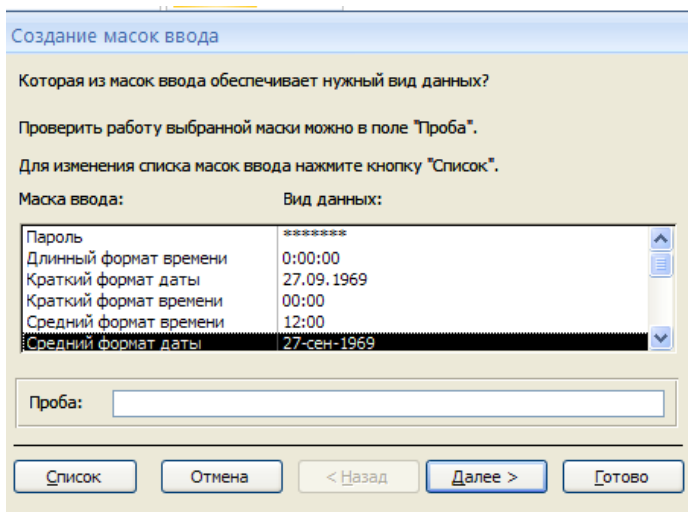


Рис. 16 Мастер створення масок введення даних у поле.

Маску введення даних можна створювати за допомогою майстра **Створення масок введення (Создание масок ввода)**. Перша сторінка цього майстра зображена на рис. 16 .

Введення даних у поля типу OLE мають деякі особливості. У поле об'єкта OLE можна вводити дані, не бачачи об'єкта. В цих полях можуть міститися різні типи даних, у тому числі растрові зображення, звукові формати, ділова графіка, Word і Excel - файли. Будь-який об'єкт, який підтримується сервером OLE, може бути збережений у полі OLE. Для того, щоб їх можна було бачити або чути, вони вставляються, зазвичай, у форми, а не в режимі таблиці. Після вставлення об'єкта OLE в поле комірка в таблиці міститиме текст про тип об'єкта. Для вставлення об'єкта OLE в поле таблиці необхідно скопіювати його до буфера обміну, а потім вміст буфера вставити в поле таблиці. Можна також відкрити контекстне меню об'єкта OLE і виконати команду **Вставити об'єкт**.

Щоб створити гіперпосилання треба спочатку у режимі конструктора ввести тип поля **Гіперпосилання (Гиперссылка)**. В режимі таблиці треба курсором вибрати рядок поля, якому буде відповідати гіперпосилання, та вибрати команду контекстного меню **Гіперссылка / Изменить гиперссылку**. Відкриється вікно **Вставка гиперссылки**, в якому треба буде вказати шлях до файлу гіперпосилання (рис. 17). Назва файлу відтвориться у вибраному рядку поля гіперпосилання.

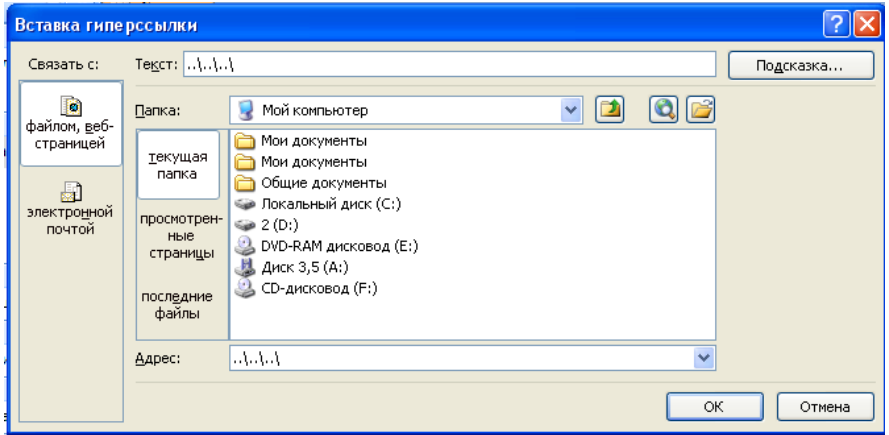


Рис. 17 Вікно Вставка гиперссылки.

Для заповнення поля MEMO натискаємо комбінацію клавіш <Shif+F2>, попередньо встановивши курсор у полі MEMO. Відкривається діалогове вікно Область вводу. Після введення або редагування даних у цьому вікні клацніть на кнопці ОК.

Аналогічним чином заповнюються інші таблиці: Академгрупи Студентів, Успішність, Дисципліни. Після заповнення таблиця Студенти може мати, наприклад, наступний вигляд.

	Студенти	Дисципліни							×
	Код студе	Код групи	Імя	Прізвище	По батькоє	рік нар.	стать	Доб	
+	1	1	Іван	Іванов	Івановіч	1995	<input checked="" type="checkbox"/>		
+	2	1	Михайло	Міхайлов	Міхайлович	1993	<input checked="" type="checkbox"/>		
+	3	1	Ольга	Ольжич	Олегівна	1994	<input type="checkbox"/>		
+	4	2	Микола	Миколаєнко	Миколаєвич	1994	<input checked="" type="checkbox"/>		
+	5	2	Марія	Миколаєнко	Миколаївна	1995	<input type="checkbox"/>		
+	6	2	Іван	Міхайлов	Івановіч	1993	<input checked="" type="checkbox"/>		
*	(№)						<input type="checkbox"/>		

Рис. 18 Приклад таблиці Студенти.

5.3. Пошук і заміна даних у таблицях

Пошук даних у таблиці великого обсягу, який виконується за допомогою кнопок переходу, може зайняти багато часу, тому для

пошуку і заміни даних в полях необхідно використовувати спеціальні інструменти.

Необхідний запис у таблиці можна знайти за значенням будь-якого її поля або за фрагментом його значення. Для цього у відкритій таблиці у режимі таблиці необхідно встановити курсор у стовпець, за значенням якого потрібно шукати запис і вибрати команду **Главная/Найти/Найти**. Відкриється допоміжне вікно **Пошук і заміна (Поиск и замена)**, зображене на рис. 19.

У полі **Зразок (Образец)** цього вікна вводиться шаблон пошуку і натискається кнопка **Найти далее**. У результаті цього курсор встановиться на рядку з цим значенням. Шаблон пошуку створюється за загальними правилами.

У полі **Совпадение** можна вибрати одне із значень, що зменшує необхідність використання метасимволів:

— **С любой частью поля** (зразок може знаходитись всередині поля, наприклад);

— **Поля целиком** (зразок без метасимволів повинен співпадати з усім значенням поля);

— **С начала поля** (будуть знайдені тільки ті поля, які починаються зі зразка).

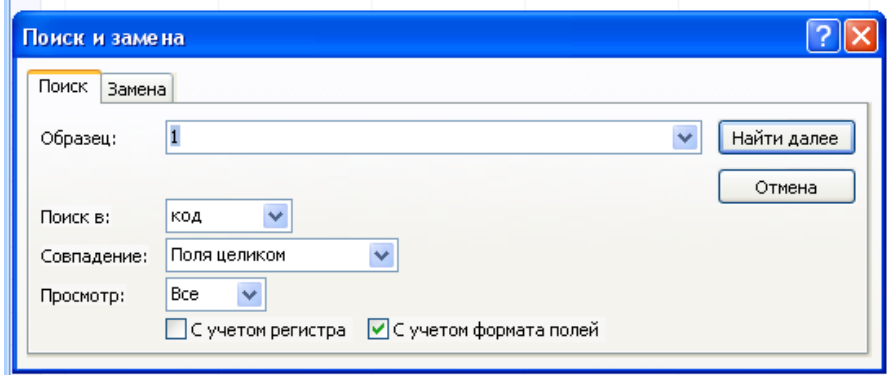


Рис. 19 Вікно пошуку і заміни.

Якщо прапорець у полі **С учетом регистра** встановлений, пошук здійснюється з урахуванням регістру, тобто зразок повинен точно відповідати фрагменту, що шукається. Якщо прапорець знятий, зразок можна вводити і великими, і малими буквами.

У разі зняття прапорця **С учетом формата полей** форматування не враховується. Наприклад, значення \$5 493 буде знайдено і для

зразка 5493. Якщо ж прапорець встановлено, то зразок повинен бути таким самим, як і шуканий фрагмент, тобто \$5 493.

Для початку пошуку слід натиснути кнопку **Знайти далі (Найти далее)**. На знайдене значення встановлюється курсор і воно висвітлюється іншим кольором. Для продовження пошуку необхідно знову натиснути кнопку **Знайти далі**. Допоміжне вікно **Пошук і заміна** можна закрити після відшукування необхідного запису.

Знайдене значення поля можна змінювати, вводити нове значення, вставляти додаткові символи. Однак слід пам'ятати, що поле типу **Лічильник**, поля, що обчислюються, заблоковані та відключені поля змінювати не можна. Зміни в полі можна відмінити тільки після першого переходу до нового поля. Після переходу до другого поля скасування виконати не можна. За аналогією зміни в даному запису можна відмінити тільки після першого переходу до нового запису.

Заміна здійснюється на вкладці **Замена** (рис. 20). Знайдений текст тут замінюється на текст з поля **Замініть на**. Однократна заміна першого входження виконується кнопкою **Замініть**, а усі входження замінюються кнопкою **Замініть все**.

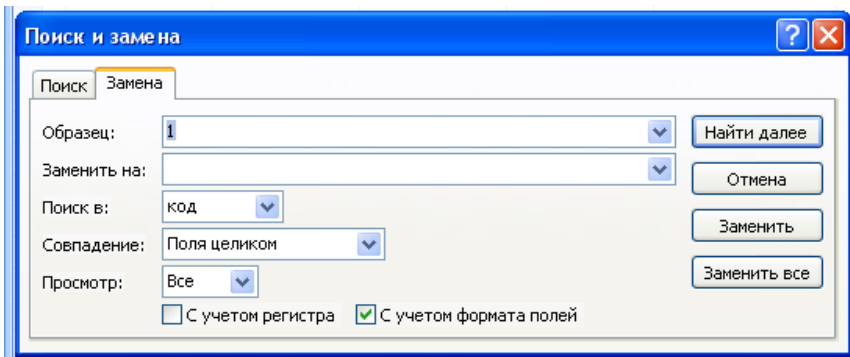


Рис. 20 Вкладка **Заміна** вікна **Пошук і заміна**.

Для швидкого пошуку першого входження зразка доцільно скористатися також текстовим полем **Пошук (Поиск)**, яке знаходиться в нижній частині вікна таблиці. Пошук завершується після першого знаходження.

Записи з таблиці можна копіювати, вирізати до буфера обміну і потім за допомогою кнопки **Вставити**, яка знаходиться у розділі **Головна / Буфер обміну**, вставляти в іншу таблицю, а також у документи Word і Excel.

Для вилучення запису необхідно клацанням Миші відмітити його (рис. 21) й натиснути клавішу **Del**, або виконати команду контекстного меню **Видалити запис (Удалить запись)**. Відкриється допоміжне вікно, де потрібно підтвердити вилучення. Для відмічання рядку треба підвести курсор у крайню ліву частину рядка до виникання чорної стрілки і натиснути ліву кнопку миші (рис. 21).

	Код студе	Код групи	Імя	Прізвище	По батькові	рік нар.	стать	Доб
+	1	1	Іван	Іванов	Івановіч	1995	<input checked="" type="checkbox"/>	
+	2	1	Михайло	Міхайлов	Міхайлович	1993	<input checked="" type="checkbox"/>	
+	3	1	Ольга	Ольжич	Олегівна	1994	<input type="checkbox"/>	
+	4	2	Микола	Миколаенко	Миколаевич	1994	<input checked="" type="checkbox"/>	
+	5	2	Марія	Миколаенко	Миколаївна	1995	<input type="checkbox"/>	
➔+	6	2	Іван	Міхайлов	Івановіч	1993	<input checked="" type="checkbox"/>	
*	(№)						<input type="checkbox"/>	

Рис. 21 Виділення запису у таблиці.

Слід пам'ятати, що в пов'язаних таблицях з встановленим прапорцем **Забезпечення цілісності даних** вилучити запис не завжди вдасться.

Іноколи виникає необхідність для кращого огляду таблиці приховати деякі стовпці. Для цього необхідно виділити певні стовпці і виконати у групі **Главная / Записи** команду **Додатково / Видобрати стовпці (Дополнительно / Отобразить столбцы)**. У вікні **Отображені стовпці** поставити «галочки» перед стовпцями (рис. 22).

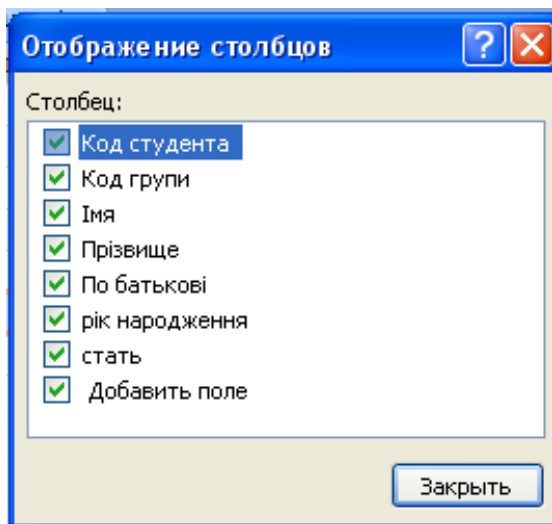


Рис. 22 Вікно для відображення і приховання стовців.

Запитання для самоконтролю знань

1. Які є способи пошуку записів у таблицях?
2. Значення яких полів змінювати неможливо?
3. Які операції можна виконувати зі знайденим записом?
4. Як можна приховати стовпці на екрані?
5. Як можна вилучити знайдені записи?

5.4. Сортування і фільтрація записів

Нагадаємо, що за замовчуванням записи таблиці виводяться на екран упорядкованими за значенням первинного ключа. Однак, часто виникає необхідність мати записи, упорядковані за значенням інших стовпців. В системі Access 2007 передбачено сортування записів за одним і за кількома стовпцями. Для сортування за одним стовпцем треба відмітити цей стовпець і натиснути кнопку **Главная / Сортировка и фильтр За зростанням (А-»Я)** або **За спаданням (Я-»А)**. При цьому справа от імені стовпця з'являється вертикальна стрілочка. Видалення сортування здійснюється командою **Главная / Сортировка и фильтр / Очистить все сортировки**. Для відмічання стовпця треба підвести курсор у крайню верхню його частину до виникання чорної стрілки і натиснути ліву кнопку миші (рис. 23).

Можна також скористатися і контекстним меню стовпця. Для сортування за кількома стовпцями треба виділити ці стовпці й скористатися одним із вказаних способів. У першу чергу сортування записів виконується за значенням лівого стовпця. Якщо в ньому є поля, які співпадають, то певні записи сортуються за значенням наступного стовпця.

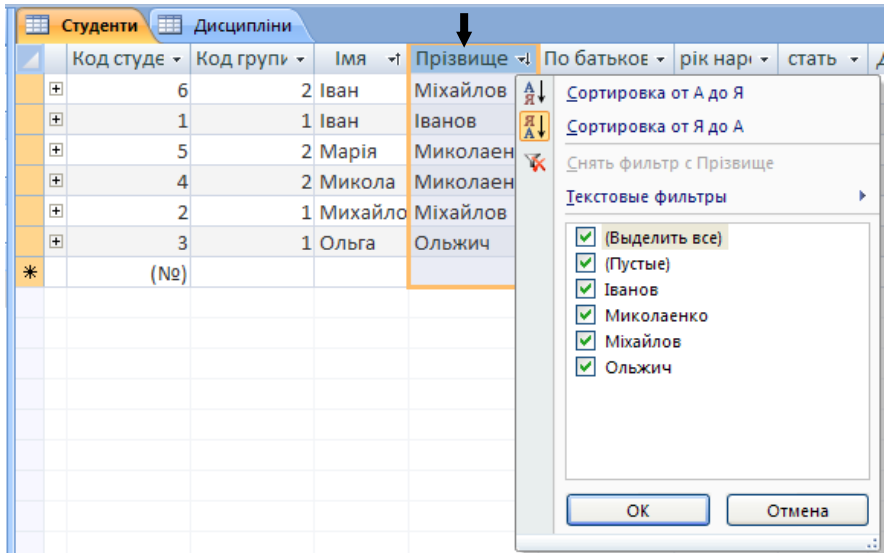


Рис.23 Виділення стовпця і його контекстне меню.

Фільтрація — це відбір із таблиці записів, які містять задане значення в обраних полях. В Access 2007 забезпечується фільтрація за виділенням і за формою. Фільтрація за виділенням — це відбір записів на основі значень поточного поля. Для її реалізації спочатку треба впорядкувати записи за значенням стовпця, який використовується при фільтрації. Потім необхідно встановити курсор на тому значенні стовпця (наприклад, X), за яким буде виконуватися фільтрація, й виконати команду **Виділення (Выделение)** у групі **Сортування й фільтр**.

Для текстової змінної відкриється меню з такими умовами: дорівнює X, не дорівнює X, містить X, не містить X. Для числової змінної відкриється меню з умовами: дорівнює X, не дорівнює X, більше або дорівнює X, менше або дорівнює X, між... . Вибіримо умову і отримаємо записи, у яких відповідне поле відповідає цим умовам.

У нижній частині таблиці праворуч від кнопки навігації висвітиться напис **С фільтром** (або **Нет фільтра**), що означає, що записи відфільтровані (або не відфільтровані). Крім того, у групі **Сортування й фільтр** міститься кнопка **Фільтр і Видалити фільтр**, за допомогою якої вмикається або вимикається фільтр. Послідовно натискаючи ці кнопки, можна бачити як змінюється вміст таблиці з фільтром і без фільтра. До відфільтрованих даних можна застосувати ще кілька фільтрів.

Більш складний фільтр вмикається командою **Расширенный фильтр** в групі **Сортування й фільтр** кнопка **Параметри расширенного фильтра**. При цьому відкривається вікно аналогічне вікну створення запитів.

Для скасування попередньо встановленого фільтру необхідно натиснути кнопку **Фільтр** у групі **Сортування й фільтр** і у меню, що відкривається, виконати команду **Видалення фільтру**....

5.5. Контрольні питання до 5 розділу

1. У якому порядку виводяться записи таблиці на екран?
2. Як можна здійснювати навігацію в записах таблиці?
3. У чому полягають особливості введення даних у поля типу OLE?
4. Якими способами можна виконати сортування записів таблиці?
5. У чому полягає сутність фільтрування записів?
6. Як виконується фільтрування записів за виділенням?
7. У чому полягає сутність фільтрування записів за формою?
8. Як здійснюється перейменування таблиць?
9. Опишіть процес створення таблиці в режимі конструктора.
10. Які типи даних можна використовувати у таблицях?
11. Якими властивостями може володіти поле?
12. Які символи можна використовувати для форматів числових і грошових полів?

6. Створення запитів.

При роботі з БД досить часто виникає проблема раціонального її використання. Одним з потужних засобів сучасних СУБД є запити, які дають можливість користувачу "**задавати питання**" базі даних. Результати запиту ("**відповідь**") завжди можна вивести на друк або монітор.

Запит (Запрос) - це вираз, який визначає яку інформацію необхідно відшукати в одній або декількох таблицях. За допомогою запиту можна також виконувати деякі операції з даними таблиць (таблиці) і здійснювати узагальнення даних. Запити можуть використовуватись як джерела інформації для форм і звітів. В цьому випадку в запиті використовуються дані з декількох таблиць. Access виконує запит кожний раз коли здійснюється відкриття форми або звіту і, як наслідок, завжди можна бути впевненим, що інформація яка виведена на зовнішній носій (монітор) завжди сама "свіжа".

В СУБД Access передбачено декілька видів запитів:

1. запит на вибірку задає запит про дані таблиці бази та видає вказаний їх динамічний набір в режимі таблиці (або форми);
2. запит на зміну використовується для зміни або переміщення даних; до цього типу належать запити на доповнення записів та їх видалення, створення чи оновлення таблиці;
3. перехресний запит здійснює групування даних та їх видачу в компактному вигляді;
4. підсумковий запит видає підсумкові значення вказаних груп даних.

Використання певного запиту залежить від конкретного завдання, яке ставить користувач при зверненні до бази даних та від потрібної форми подання даних.

6.1. Створення запиту на вибірку

Цей тип запиту є основним, він найчастіше використовується у роботі. З нього доцільно починати створення інших типів запитів.

Створення запиту здійснюється за допомогою інструментів: **Майстер запитів** або **Конструктор запитів**, які розташовані у групі **Создание / Другие**.

Самий простий спосіб створення Запитів це використання Майстра. Він задає питання і автоматично будує Запит. Але Метод конструктора більш наочний і дозволяє вникнути у суть питання.

При використанні режиму **Конструктор** програма відкриває вікно **Запрос1**, а в ньому вікно **Добавление таблицы**. У останньому вікні заданий перелік таблиць поточної бази, з яких командою **Добавить** визначаються ті таблиці, дані з яких будуть відтворені у запиті. Вказані таблиці з переліком їх полів виводяться у верхню частину вікна **Запрос1**.

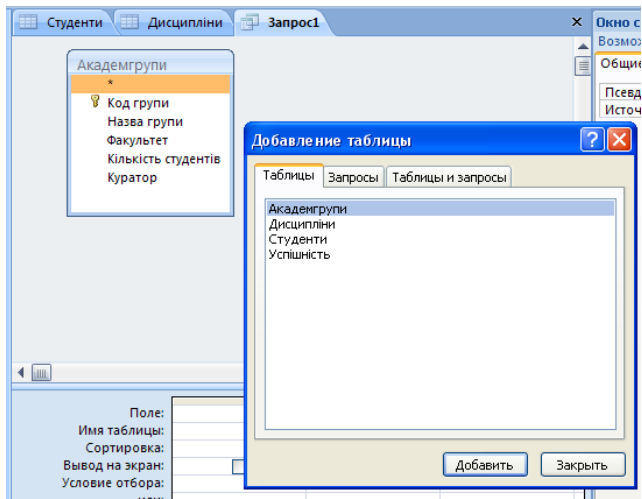


Рис. 24 Вікно створення запитів.

Якщо зв'язки таблиць були вже визначені, ці зв'язки будуть вказані у вікні. Якщо в таблицях, що додаються у запит, поля первинних та чужих ключів введені, програма сама встановить зв'язок між таблицями. Але треба цей зв'язок уточнити. Закінчивши вибір таблиць, вікно **Добавление таблицы** треба закрити.

Далі робота ведеться у вікні **Запрос на выборку**. Зовнішній вигляд вікна показаний на рис. 24.

В нижній частині вікна у рядок **Поле** у стовпці бланку запиту в бажаному порядку заносяться назви полів таблиць, які мають попасти у запит. Операцію можна здійснити різними способами, але найзручнішим є вибір назви поля у списку, який можна розкрити при встановленні курсору у рядок **Поле** відповідного стовпця. Це забезпечує крім запису імені поля одночасний автоматичний запис імені таблиці у рядок **Имя таблицы**, завдяки чому виключається плутанина в разі, коли різні таблиці мають поля з однаковими іменами.

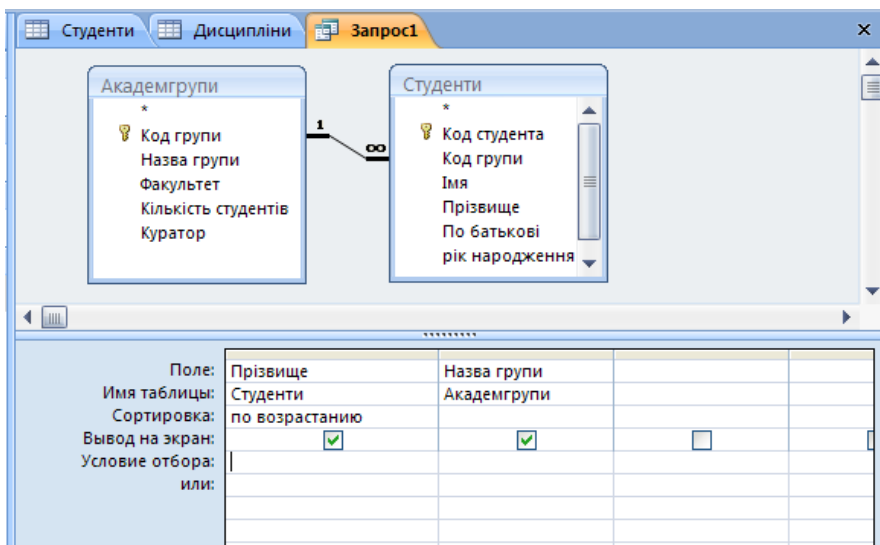


Рис. 25. Вікно створення запиту у режимі Конструктор

. Назви полів можуть бути також просто перетягнуті мишею в потрібний стовпець бланка із списку полів у верхній частині вікна. За необхідності назви полів у створюваному запиті можна змінити, використовуючи **Вікно свойств**, у якому **Свойства поля** рядок **Подпись** містить нову назву. Зміна імені відбудеться у запиті при його відкритті. У цьому ж вікні можна встановити інший (порівняно з табличним) формат поля.

6.2. Сортування і умови відбору даних

Для зручності перегляду записів у запиті можна встановити порядок сортування записів. У рядку **Сортировка** (рис. 26) можна вибрати зі списку порядок сортування **По возрастанию** або **По убыванию**.

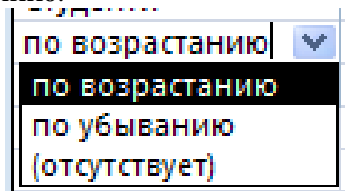


Рис. 26 Список сортування.

Наступним кроком є завдання умов відбору записів. Запис умов відбору здійснюється у вигляді умов на значення виразів. Ці умови і вирази зручно будувати за допомогою Будівника виразів (**Построитель выражений**), який відкривається командою контекстного меню **Побудувати...** або кнопка **Конструктор / Настройка запроса / Построитель**.

Будівник виразів має вигляд, як зображено на рис. 27 .

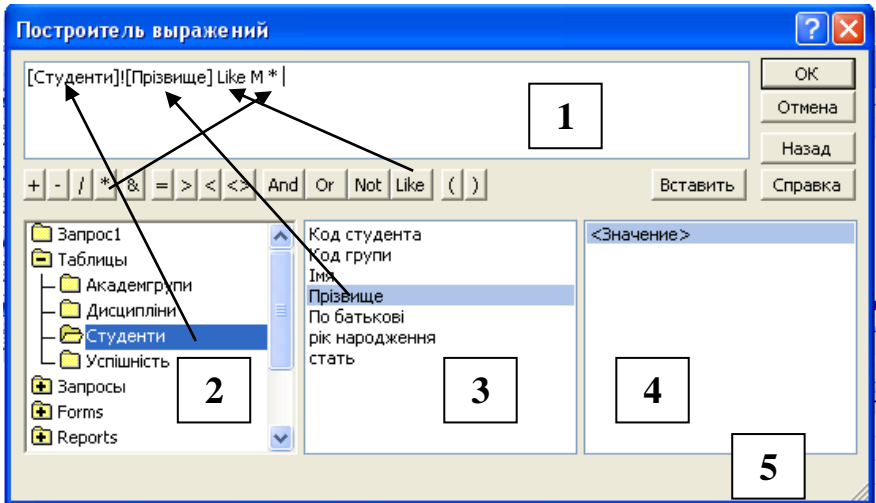


Рис. 27 Вікно будівника виразів.

Цифрами на малюнку позначено:

1. Поле виразу. Тут можна ввести вираз і додати його елементи, двічі клацнувши їх у доступних нижче списках елементів.

2. Об'єкти БД. Клацніть об'єкт, щоб переглянути його складові в наступному списку.

3. Складові об'єктів. Це поля таблиць або категорії вбудованих об'єктів. Клацніть

4. Значення. Склад обраної категорії. Двічі клацніть значення, щоб додати його в поле вираження.

5. Довідка та відомості про виділений рядок. Клацніть це посилавання (якщо вона доступна), щоб переглянути статтю довідки про виділене значення виразу.

Математичні вирази складаються зі змінних, констант, функцій, полів і математичних операцій, дужок. Логічні вирази складаються з

математичних операцій і операцій порівняння, а також логічних виразів, з'єднаних логічними операціями.

Для задання шаблону пошуку текстових виразів використовується операція **Like**. В шаблоні також використовуються знаки заміни перелічені у додатку 1. Наприклад для створення запиту «Прізвище на М» умови створені добудовником виразів мають вигляд :

[Студенти]![Прізвище] Like "М*"

Це означає, що треба провести пошук по полю **Прізвище** у таблиці **Студенти** так, щоб перша буква прізвища була буква «М», а усі наступні будь-які, про це говорить знак «*». На рис. 28 представлені конструктор цього запиту і результати його роботи.

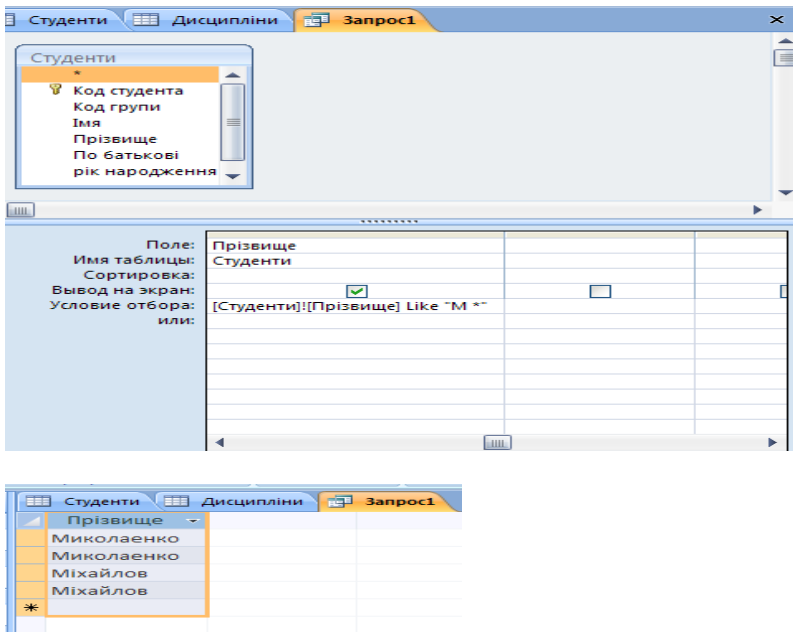


Рис. 28 Конструкція запиту «Прізвище на М» і результат його виконання (унизу).

Можна створювати і інші умови як для цього поля, так і для інших, і з'єднувати їх логічними операціями. Умови в одному рядку об'єднуються операцією AND, а умови у різних рядках операцією OR (рис. 29).

Також в умові можна використовувати оператори OR, AND та NOT, та дужки.

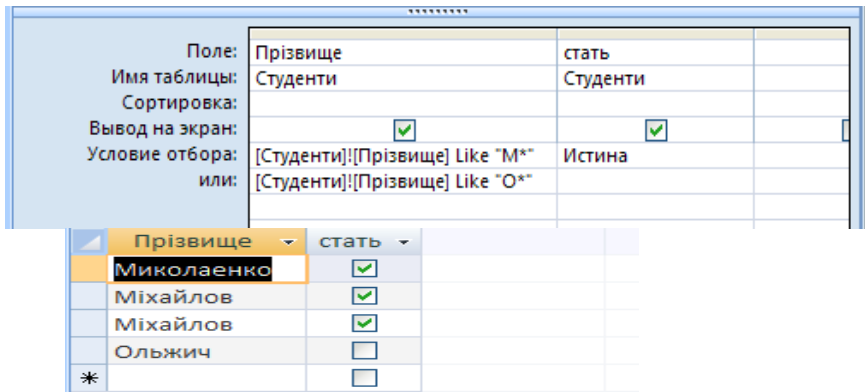


Рис. 29 Складні умови і результати роботи запиту.

Параметричний запит.

У програмі MS Access передбачена ситуація, коли умови відбору точно невідомі під час створення запиту. В цьому випадку корисним є використання параметру – елементу, який містить в собі змінне значення, що впливає на результати відбору. Параметр може мати тип, допустимий для MS Access. Параметр записується у рядку **Умовие отбора** у квадратних дужках бажано зрозумілою назвою. При відкритті запиту з'являється додаткове діалогове вікно (запит значення параметра) з назвою параметра. Після введення конкретного значення параметру відкривається запит, який виведе записи, що відповідають цьому значенню (рис. 30).

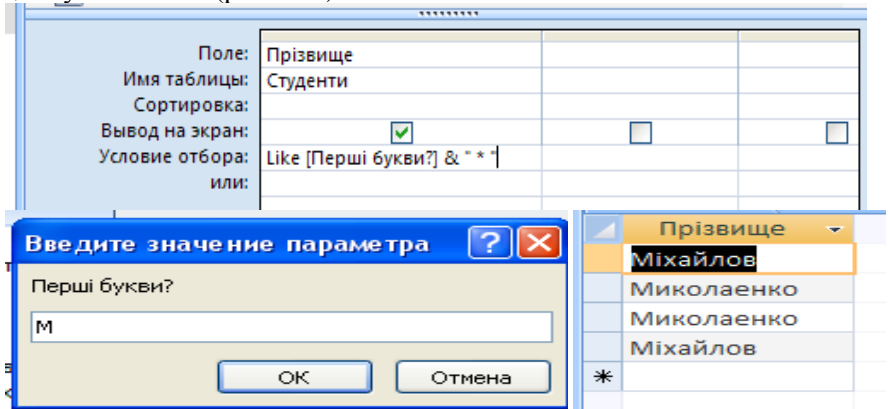


Рис. 30 Конструкція параметричного запиту, вікно введення параметру і результат роботи запиту..

На рис. 30 зображений конструктор запиту з умовою пошуку Прізвища студентів, у яких перші букви представлені параметром [Перші букви], а останні будь-які. При виконанні запиту значення параметру запитується у вікні **Введіть значення параметра**. При введенні букви М отримуємо результат, що представлений на рис. 30.

Нижче подано перелік операцій і функцій, які найчастіше використовуються при формуванні умов запитів.

Знаки арифметичних операцій: \wedge – ступінь, $*$ – множення, $/$ – ділення, \backslash – ділення з одержанням цілого значення, $+$ – плюс, $-$ – мінус.

Оператори порівняння: $>$ – більше, $>=$ – більше дорівнює, $=$ – дорівнює, $<$ – не дорівнює.

Логічні оператори: **And** – і, **Eqv** – еквівалент, **Not** – ні, **Or** – або.

Функції: **Sum** – визначає суму значень даного поля для всіх записів, відібраних запитом; **Avg** – середнє значення; **Min**, **Max** – відповідно мінімальне, максимальне значення; **Count** – кількість відібраних записів; **First**, **Last** – перше, останнє значення.

Повний перелік операторів і функцій можна отримати, користуючись файлом довідки.

6.3. Обробка даних у запиті

При формуванні запитів досить часто виникає потреба розрахунку нового показника. Важливою рисою запиту є можливість виконання різноманітних математичних дій над даними полів, чим забезпечується потрібна обробка даних. Математичні вирази розміщуються у нових полях запиту, а результат відтворюється в них при виведенні запиту. Створення виразів зручно виконувати за допомогою майстра **Побудувати виражений**, який викликається кнопкою **Побудувати...** після вказання курсором обраного стовпця бланку запиту. Команду **Побудувати** можна також задати, якщо викликати контекстне меню, встановивши курсор у стовпець, де необхідно провести обчислення, та натиснувши правий клавіш. Майстер надає користувачу широкий набір вбудованих функцій, який відкривається кнопкою **Функції\Встроенные функции**. Побудова математичних виразів виконується за правилами, стандартними для всіх офісних програм.

Майстер надає полям, де введені математичні вирази, стандартні імена (Выражение N). Доцільно це ім'я у рядку Поле видалити та записати більш змістовне ім'я. Це виконується у вікні **Свойства поля**, яке відкривається у режимі конструктора.

Наприклад розрахуємо вік студентів з прізвищем на певну букву. Для цього треба від поточного року відняти рік народження. Цій вираз створимо у будівнику виразів (рис. 31). Результат роботи цього запиту представлений на рис. 31 .

Поле:	Прізвище	Выражение1: Year(Date())-[Студенти]![рік народження]
Имя таблицы:	Студенти	
Сортировка:		
Вывод на экран:	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Условие отбора:	Like [Перші букви?] & "*"	
или:		

Прізвище	Выражение1
Міхайлов	20
Миколаенко	19
Миколаенко	18
Міхайлов	20

Рис. 31 Будівник виразів і результат роботи запиту.

На практиці може виникнути необхідність виконати певні дії над всіма даними певного поля. Наприклад, якщо треба одержати дані про загальну кількість студентів в усіх групах. Для цього використовується підсумковий запит. Він створюється точно так же, як і запит на вибірку. Через кнопку панелі інструментів **Конструктор** / **Показати** **или скрити** / **Итоги** (або через контекстне меню) відкрити ще один рядок: **Групповая операция** (рис. 32).

Поле:	Прізвище	стать	[Студенти]![Прізвище]
Имя таблицы:	Студенти	Студенти	
Групповая операция:	Группировка	Группировка	Группировка
Сортировка:	по возрастанию		
Вывод на экран:	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Условие отбора:		Истина	Like "M**"
или:			Like "O**"

Рис. 32 Конструктор з полем для виконання групових операцій.

У бланк запиту заносяться поля, у яких потрібно виконати групові операції. У кожному полі у рядку **Групповая операция** з'являється запис **Группировка**. Розкривши у цьому рядку список,

можна з переліку вибрати конкретну операцію, назва якої буде відтворена у рядку. До списку входить дев'ять функцій: Sum (сума), Avg (середнє арифметичне), Min, Max (мінімальне та максимальне значення), Count (Кількість записів), StDvg (стандартне відхилення), Var (дисперсія), First, Last (перше та останнє значення в полі). При виборі конкретної функції треба враховувати тип даних, над якими вона буде виконуватися.

Після переходу до режиму таблиць буде виведений підсумковий запит. До речі, в режимі Конструктор, викликавши вікно Свойства, можна змінити назви полів (рядок Подпись), надані програмою, на більш зрозумілі. Після розробки бланка запиту в режимі Конструктор вікно **Запит** закривається. Програма запитує про необхідність збереження змін у вікні, пропонує ім'я запиту та, одержавши відповіді, створює піктограму в області переходів. Його можна переглянути, надавши команду **Открыть**.

Відкривши запит у режимі таблиці, можна здійснити певні зміни щодо його зовнішнього вигляду. Для цього використовуються команди пункту меню Формат: Шрифт..., Режим таблиць..., Высота строки..., Ширина столбца... і т.д., зміст яких є повністю зрозумілим.

Запити можна створювати на ґрунті багатьох основних таблиць і представляти результати у вигляді однієї таблиці. Зміна даних у таблиці Запиту завжди відтворюється в основній таблиці даних.

Запит на вибірку є основою для створення запитів інших типів. Вибрати тип запиту можна у меню **Конструктор** у розділі **Тип за-проса**.

6.4. Мова запитів SQL

SQL (англ. Structured query language — мова структурованих запитів) — декларативна діалогова мова програмування для взаємодії користувача з базами даних, що застосовується для формування запитів, оновлення і керування реляційними БД.

Вона використовується для здійснення запиту і внесення змін до бази даних, а також управління базами даних. Багато баз даних підтримує SQL. Будь-який запит створений у конструкторі або іншим способом можна переглянути у режимі SQL (рис. 33)

SQL не залежить від конкретної СУБД. Тексти SQL-запитів можуть бути досить легко перенесені з однієї СУБД в іншу.

Наявність стандартів (міжнародного стандарту ANSI SQL-92) і набору тестів для виявлення сумісності і відповідності конкретній ре-

алізації SQL загальноприйнятому стандарту сприяє використанню мови.

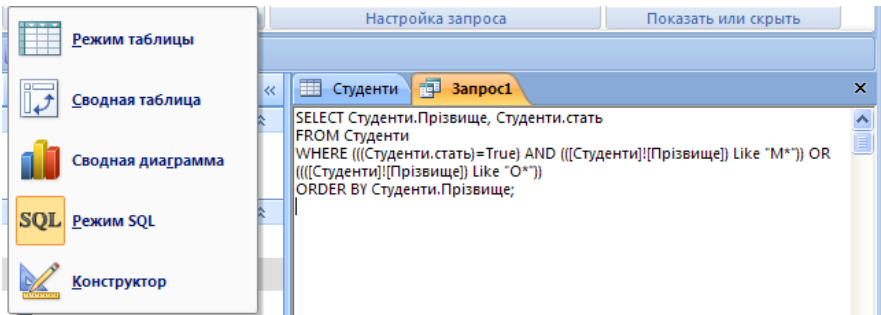


Рис. 33 Видяг запиту у режимі SQL.

За допомогою SQL програміст указує тільки, які дані потрібно витягнути або модифікувати. Те, яким чином це зробити, вирішує СУБД безпосередньо при обробці SQL-запиту.

Недоліком мови є її невідповідність реляційної моделі даних. SQL не є істинно реляційною мовою.

SQL створювався, як засіб роботи кінцевого користувача, але він став настільки складним, що перетворився на інструмент програміста.

Оператори мови SQL.

Оператор **SELECT** дозволяє вибирати дані з бази. Загалом оператор **SELECT** виглядає так:

```
SELECT імена_полів  
FROM імена_таблиць  
WHERE умова;
```

Імена полів записуються через кому. Якщо потрібно вибрати всі поля, пишуть зірочку ("*"). При потребі можна уточнити з якої таблиці брати поле, додавши перед його іменем.

SELECT може видати нам рядки що повторюються. Якщо ми хочемо мати тільки унікальні значення, то можемо уникнути повторень командою **DISTINCT**.

```
SELECT DISTINCT поля FROM таблиці;
```

Умови відбору даних створюються так, як у конструкторі.

Також до запиту **SELECT** можна додати команду **ORDER BY**, що дозволяє впорядкувати результат за заданими стовпцями. Щоб сортувати в зворотному порядку після стовпців, за якими сортують, пишуть **DESC**:

```
SELECT name FROM students ORDER BY name DESC;
```

З'єднання використовуються для запитів з кількох таблиць, що базуються на зв'язках між певними стовпцями таблиць.

Для опису з'єднання таблиць використовується INNER JOIN (або JOIN), де є хоч одне співпадіння в стовпцях таблиці. Наприклад:

```
SELECT назви_стовпців  
FROM перша_таблиця  
INNER JOIN друга_таблиця  
ON перша_таблиця.назва_стовпця=друга_таблиця.назва_стовпця
```

І це буде те ж саме що і

```
SELECT назви_стовпців  
FROM перша_таблиця, друга_таблиця
```

WHERE перша_таблиця.назва_стовпця = друга_таблиця.назва_стовпця

LEFT JOIN працює та пишеться майже так само, але повертає таблицю, в яку входять всі записи лівої таблиці (недостаючі записи з правої заповнюються позначками NULL). RIGHT JOIN навпаки. FULL JOIN повертає об'єднання результатів RIGHT та LEFT JOIN.

Оператор **INSERT** додає до таблиці рядок. Має такий синтаксис:

```
INSERT INTO назва_таблиці VALUES (список_значень);
```

Значення мають йти в такому ж порядку, як і стовпці таблиці. При необхідності можна задати конкретні стовпці, та конкретні значення:

```
INSERT INTO students (name) VALUES ('Іван');
```

Всі інші поля отримають значення за замовчуванням.

UPDATE змінює значення полів в уже існуючих записах. Синтаксис:

```
UPDATE назва_таблиці SET стовпець1=значення1, стовпець2=значення2, ... WHERE умова;
```

Якщо забути задати умову, то зміняться всі записи таблиці.

DELETE найпростіший оператор:

```
DELETE FROM назва_таблиці WHERE умова;
```

Команди "Відмінити" у цьому випадку немає.

Усі запити, що створені у конструкторі можна переглянути на мові SQL. Для цього

6.5. Контрольні питання до 6 розділу

1. Які бувають типи запитів?
2. Для чого служить запит на вибірку?
3. Що утворюється в результаті виконання запиту?

4. Якими способами можна створювати запити?
5. Що таке параметричний запит? Якого типу він може бути?
6. Що таке критерій відбору?
7. Опишіть процес створення запиту в режимі конструктора.
8. Для чого служить рядок Условие отбора у вікні конструктора запиту?
9. В яких режимах можна переглядати запит? Як перейти з одного режиму перегляду запиту в інший?
10. Що таке будівник виразів?
11. Що таке обчислювальне поле? Як його створити?
12. Чи можна відредагувати створений запит? Як?
13. Як можна об'єднувати умови вибору? У яких випадках використовується логічне І, а у яких - логічне АБО?
14. Як можна об'єднати записи декількох таблиць?
15. Які є оператори мови SQL?

7. Форми та робота з ними

Форми є зручним інтерфейсом і допомагають користувачам при введенні та редагуванні даних. У режимі форми більш повно можливо скористатися засобами, що надаються Windows (різні шрифти, кольори, графіка і т.д.). З допомогою форми можна відобразити всю інформацію, що міститься в кожному записі, у той час як в режимі таблиці частина полів може виходити за межі екрану.

Через форми може здійснюватися введення даних у взаємопов'язані таблиці бази даних, перегляд даних, а також їх зміну. Працюючи з формою, користувач може додавати, видаляти та змінювати записи таблиць, отримувати розрахункові дані. В процесі роботи може здійснюватися контроль даних, що вводяться, встановлюватися обмеження на доступ до даних, виводитися необхідні повідомлення. Форми є основою розробки діалогових програм користувача для роботи з базою даних.

Таким чином, форма є основним засобом організації інтерфейсу користувача і для організації управління базою даних під час роботи з базами даних. Завдяки використанню зручного розташування даних, різних кольорів, шрифтів та форматів записи, що виводяться, форми стають більш наочними, внаслідок чого краще сприймаються подані в них дані.

Дані можна вводити безпосередньо в таблиці або за допомогою запитів. Але за допомогою форми це можна робити значно простіше і зручніше. Найчастіше форми для введення розробляються для таблиць, а джерелом даних форм для виведення є запити. За допомогою форми перевіряються дані, що вводяться в комірки її полів, й у разі виникнення помилок автоматично повідомляється про це користувача. За допомогою форм можна змінювати дані одночасно у кількох взаємопов'язаних таблицях.

Форма може складатися з багатьох об'єктів та елементів, які називають елементами управління. Прикладами таких елементів є прямокутник, лінія, текстове поле, прапорець, кнопка та інші. Кожен елемент керування, а також сама форма мають атрибути (параметри), наприклад, товщина лінії, колір фону, розмір шрифту тощо.

7.1. Створення форми

Розглянемо спочатку порядок створення простої форми (на основі таблиці або запиту). Найбільш простим є автоматичне створення форми. Цим засобом створюється форма, яка відображає усі поля одного поточного запису. Автоматичне створення форми на основі активної таблиці або запиту виконується командою **Создание / Формы / Форма**. Форма, яка створюється, виводиться в режимі макету, у якому можна одночасно переглядати дані й редагувати елементи керування. У нижній частині форми розміщені елементи навігації, за допомогою яких можна перейти до будь-яких записів таблиці. Якщо між таблицями встановлений зв'язок **один-ко-многим** створюється складна форма з визначенням основних та підпорядкованих полів.

Зберігаємо форму, для чого на панелі швидкого доступу виконуємо команду **Зберегти**. У вікні, яке відкриється, вводимо ім'я форми і натискаємо кнопку ОК. В області переходів з'явиться це ім'я. Закриваємо форму. Після цього форму можна використовувати у будь-який час, для чого встановити курсор на ім'я форми і двічі натиснути кнопку миші.

Засіб **Розділена форма** на основі таблиці або запиту, виділеного (або відкритого) на панелі переходів, створює форму, де дані одночасно подаються у вигляді кількох записів таблиці й у вигляді набору елементів керування одним записом. Як і за попереднім способом, форма створюється автоматично.

Форма **Кілька елементів** має кілька записів таблиці. Форма створюється на основі таблиці або запиту, виділеного в області переходів. Зовнішньо форма схожа на таблицю. За замовчуванням форма виводиться в режимі макета. У режимі конструктора до неї можна додавати додаткові елементи управління.

Якщо вигляд автоматично створеної форми не буде повністю задовольняти потребам користувача, завжди можна здійснити її корегування в режимі **Конструктор**. Конструктор форм є найпотужнішим інструментарієм створення форм.

Конструктор (рис. 34) відкривається командою **Головна / Режими / Конструктор**.

Заголовок форми			
Академгрупи			
Область даних			
Код групи:	Код гр	Кількість студентів:	Кількість
Назва групи:	Назва групи		
Факультет:	Факультет		
Куратор:	Куратор		
Таблица Студенти			
Примечание форми			

Рис. 34 Вигляд форма у режимі Конструктор.

7.2. Конструктор форм

У режимі конструктор з'являється пуста форма, на яку треба встановити елементи управління, які розташовані на панелі інструментів (рис. 34). Панель елементів управління з'являється у меню **Конструктор** коли форма відкривається у режимі конструктора.

Елементи управління поділяються на такі три групи:

1. Елементи управління, що приєднуються. Ці елементи приєд-

нуються до полів таблиці. Якщо користувач вводить дані в такий елемент, автоматично оновлюється вміст поля в поточному записі. Такий елемент можна приєднати практично до будь-якого типу даних.

2. Вільні елементи управління. Використовуються для зберігання значень, що вводяться, але не оновлюють його в полі таблиці. Використовуються для виведення текстових приміток і елементів оформлення (ліній, прямокутників), а також для збереження вільних об'єктів OLE (наприклад, малюнків), які розміщені не в таблицях, а у формі.

3. Елементи управління, що обчислюються. За їх допомогою виводиться результат обчислення математичного виразу. У виразі, у якості змінних, використовуються значення полів таблиці. Значення поля таблиці не оновлюється.

Елемент **Надпись** використовується для занесення на форму якихось постійних текстів, наприклад, назви форми і елементів управління.

Елемент **Поле** використовується для відображення змісту полів таблиць або результатів розрахунків, а також для введення даних у таблицю. Поле (1) розміщується на формі спільно з відповідним надписом (2), який містить ім'я поля (рис. 35). Надпис можна видалити. Поле за умовчужанням має ім'я Поле0.



Рис 35 Елемент Поле (1) створюється спільно з надписом (2).

Інструмент **Кнопка** дозволяє розмістити на формі командну кнопку, що буде запускати макроси або процедури VBA. Після розміщення кнопки на формі запускається майстер, який зв'яже кнопку з макросом або процедурою. На рис. 36 зображено створення кнопки для відкривання іншої форми.

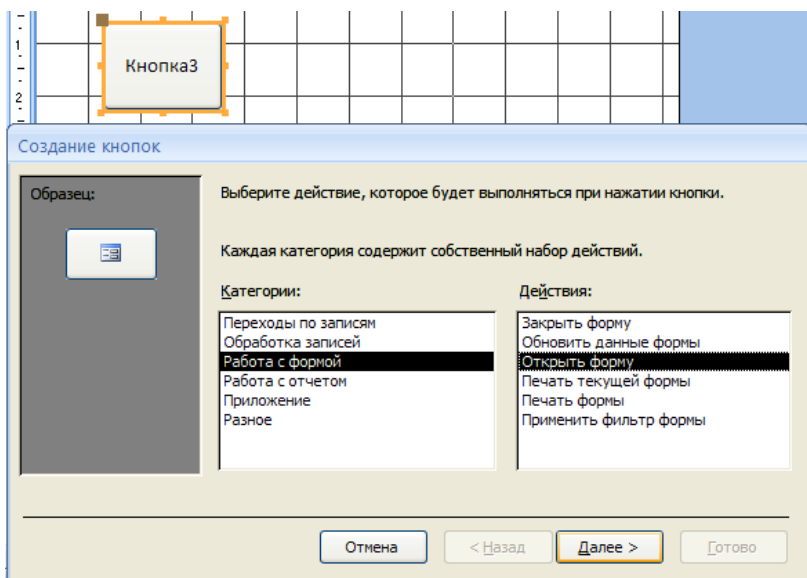


Рис. 36 Створення кнопки для відкриття іншої форми.

Елемент **Список** дозволяє вибрати зі списку будь-яке значення і занести його у таблицю. Елемент **Поле со списком** виконує такі ж дії але список відчиняється спеціальною кнопкою праворуч.

Елемент **Рисунок** містить графічний об'єкт, який розміщений у базі даних.

Присоединенная рамка объекта використовується для відображення мультимедійних файлів (графіка, звук, відео, текст, таблиці тощо), які зберігаються окремо від файлу бази даних.

Подальша робота спрямована на відпрацювання остаточного зовнішнього вигляду форми: встановлюються бажані розміри елементів, назви вкладок змінюються на більш змістовні, виділяються групи елементів за допомогою інструментів **Прямокутник** та **Лінія** і т.д.

Командою **Конструктор / Сервис / Страница свойств** відчиняється **Вікно властивостей** активного (виділеного) елемента або форми. На рис. 38 зображено вікно властивостей форми з ім'ям «Форма», що відображено у верхній частині вікна. Далі розташовані вкладки з властивостями. На вкладці **Данные** розташовані властивості, які керують даними. Перший рядок **Источник записей** показує на основі якої таблиці створена ця форма. Властивості на вкладці **Макет** відповідають за форматування елемента управління або самої форми.

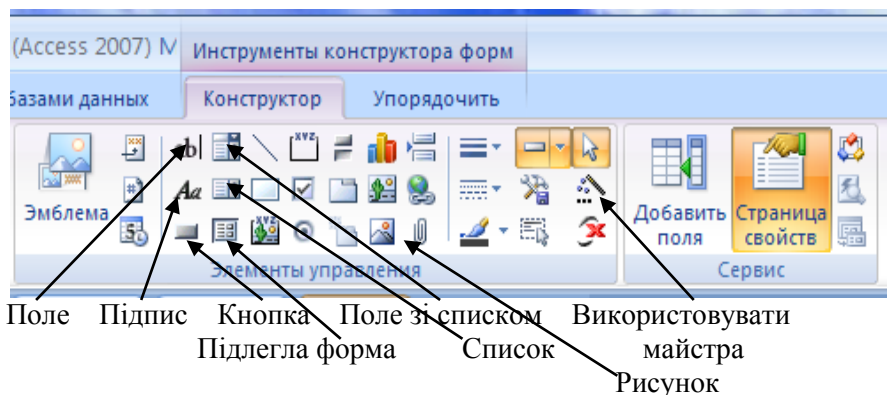


Рис. 37 Панель елементів управління.

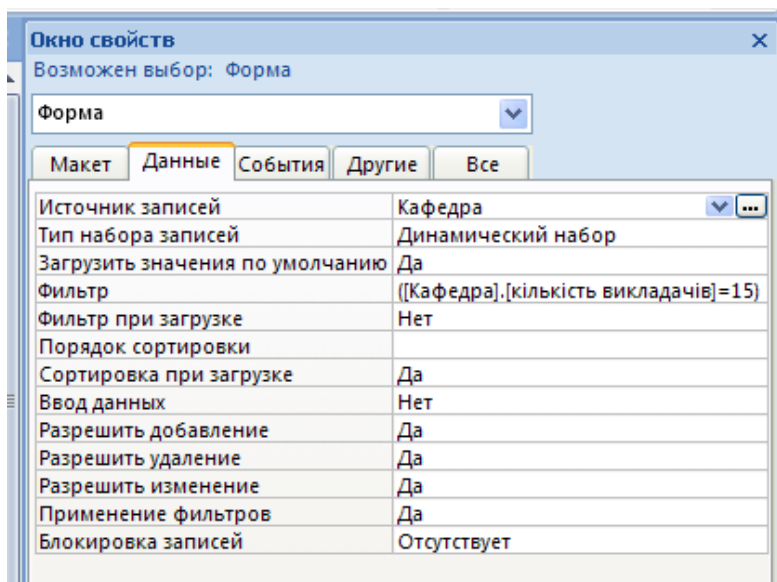


Рис. 38 Вікно властивостей форми.

7.3. Налагодження форми

Налагодженням форми, створеної майстром або конструктором, є її доопрацювання з метою надання їй більш зручного, з точки зору

конкретного користувача, зовнішнього вигляду. Налаштування здійснюється у режимі **Конструктор**. При цьому можна здійснити:

- зміну на формі шрифтів, їх розмірів та кольорів;
- зміну полів відповідно до розмірів даних, що в них відображаються;
- зміну розміщення полів на формі;
- додавання у форму графіки для оформлення груп полів;
- використання кольорового оформлення фрагментів форми;
- завдання форматів відображення даних.

Корекція виконується з використанням стандартних прийомів, прийнятих для офісних програм. Тому просто надамо певні рекомендації щодо виконання операцій.

Під час роботи з полями (взагалі з елементами управління) доцільно використовувати відомості, які подаються про них у **Вікні властивостей** (відкривається командою **Конструктор / Сервіс / Страница свойств**). Тут, зокрема, показані розміри полів та координати їх розміщення на формі.

При роботі з полями слід враховувати, що їх можна переміщувати або разом з їх надписами, або окремо від них, а також змінювати їх розмір (рис. 39). Варіант визначається виглядом курсору при виділенні поля. Якщо курсор має вигляд хрестоподібних стрілок на межі поля (1), виконується переміщення за першим варіантом. При підведенні курсору до темного прямокутника на виділеному елементі (2) поле та його надпис можна переміщувати по формі окремо.

Зміна розміру елементу здійснюється перетягуванням курсором вузлів на виділеному елементі управління (рис. 39, стрілки 3, 4,5).

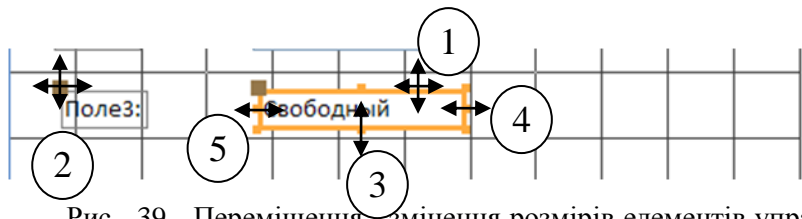


Рис. 39 Переміщення і змінення розмірів елементів управління на формі.

При зміні розмірів або розміщення елементів на формі рекомендується користуватися клавішами переміщення курсору при одночасному натисненні клавіша Ctrl або Shift. Це дозволяє виконувати потрібні операції більш точно.

Для вирівнювання елементів на формі вибирається команда з переліку, що відкривається у контекстному меню **Вирівняти**. Нагадаємо, що для одночасного виділення декількох елементів треба тримати натиснутою клавішу Shift.

Для виділення на формі груп елементів можна використовувати лінії та прямокутники, які входять до панелі елементів. Колір та товщина ліній вибираються із переліків, що розкриваються у меню **Конструктор** / **Елементи управління** кнопками **Цвет линии**, **Тип линии**, **Толщина линии**.

Частину форми, обмежену прямокутником, можна виділити іншим кольором, використовуючи кнопку **Шрифт** / **Цвет заливки/фона**.

Для створення на формі заголовку можна використати елемент **Надпись** панелі елементів.

На формі можна встановити спеціальні формати для даних, що виводяться у полях. Для цього поле виділяється та відкривається **Вікно свойств**.

Для прискорення вводу даних у форму можна використовувати поля, оформлені як поле із списком, якщо набір значень даних відомий заздалегідь. У режимі конструктора слід виділити поле, що підлягає заміні, та видалити його разом із надписом. На панелі елементів натиснути кнопку **Використовувати майстра** / **Использовать Мастера** (якщо вона не натиснута за замовченням), потім кнопку **Поле со списком** та створити на форму потрібне поле зі списком.

Майстер відкриє перше діалогове вікно (рис. 40), де треба визначити джерело значень для поля (наприклад, **Будет введен фиксированный набор значений**). У наступному вікні слід задати кількість стовпців, відрегулювати їх ширину та заповнити рядки стовпців фіксованими значеннями. Далі виконати запити майстра у наступних вікнах щодо назви поля та вийти з режиму майстер. Тепер під час занесення у форму нових записів, розкриваючи список у полі, можна просто вибирати потрібні значення даних.

Використання форми для одержання та зміни даних виконується в режимі **Форми**, який відкривається кнопкою **Открыть** або до якого можна перейти з режиму **Конструктора**. Тут здійснюється перегляд даних, їх корекція, додавання та видалення. Форми можна також використовувати для друкування інформації.

Для роботи з формою користуються лінійкою навігації унизу форми, де вибираються номери потрібних записів. Вибравши на лінійці кнопку із зірочкою, можна відкрити новий пустий запис та додати в

нього потрібні дані. Змінення у записах та нові записи будуть відтворені у полях таблиць, покладених в основу форми.

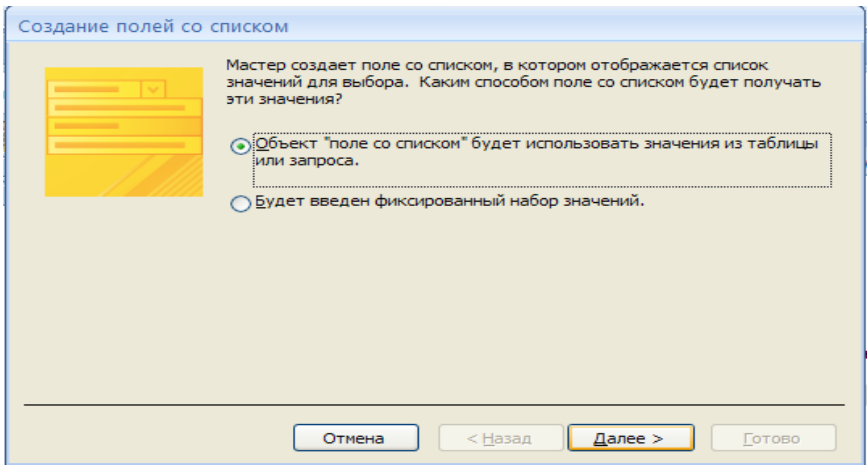


Рис. 40 Майстер створення поля зі списком.

7.4. Типи полів

У формах використовують елементи управління, які відображають дані, виконують дії та надають можливості перегляду й роботи з інформацією, що вдосконалює інтерфейс користувача (підписами, зображеннями тощо). Access підтримує три типи полів: приєднані, вільні та обчислювані.

Приєднаний елемент управління. Це елемент, джерелом даних якого є поле таблиці або запиту. Приєднані елементи використовуються для відображення значень, які отримуються з полів бази даних. Значення може бути текстом, числом, значенням «Так/Ні», зображенням або схемою. Текстове поле — найпоширеніший тип приєданого елемента управління. Наприклад, текстове поле форми, в якому відображається прізвище працівника, може отримувати дані з поля «Прізвище» таблиці «Працівники».

Вільний елемент управління. Елемент, який не має джерела даних (поля або виразу), називається вільним. Вільні елементи використовуються для відображення відомостей, ліній, прямокутників і зображень. Наприклад, підпис, в якому відображається назва звіту, є вільним.

Обчислюваний елемент управління. Це елемент управління, джерелом даних якого є вираз, а не поле. Значення, яке слід використовувати як джерело даних для цього елемента, вказується через вираження виразу. Вираз може бути комбінацією операторів (наприклад =, +), імен елементів управління, імен полів, функцій і констант. Наприклад, нижченаведений вираз обчислює вартість зі знижкою 25 відсотків: значення поля «Ціна» множиться на константне значення (0,75).

$$= [\text{Ціна за одиницю}] * 0,75$$

Вираз може використовувати дані з поля базової таблиці чи запиту або з елемента управління у формі.

Під час створення форми доцільно спершу додати й розташувати всі приєднані елементи. Потім можна додати вільні та обчислювані елементи, скориставшись засобами з групи **Елементи управління** на вкладці **Конструктор**.

Можна створити елемент управління, приєднаний до вибраного поля, перетягнувши поле з області **Список полів** до форми. В області **Список полів** відображаються поля базової таблиці або базового запиту. Для відображення області **Список полів** на вкладці **Конструктор** у групі **Сервіс** виберіть елемент **Додати поля**.

Елемент управління приєднується до поля таблиці через **Вікно властивостей** поля. Для цього ввести ім'я поля в самому елементі управління або в полі значення **Джерело записів** (Источник записей, ControlSource) у вікні властивостей.

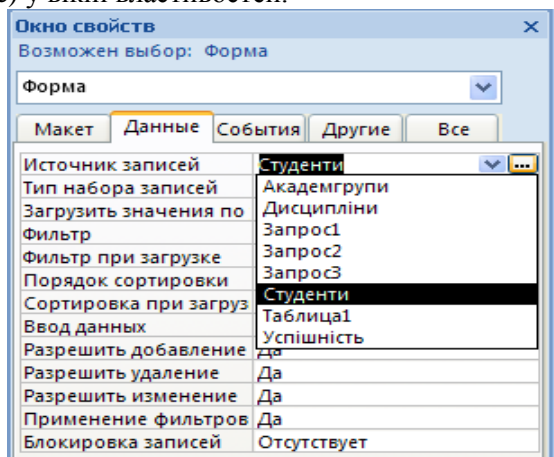


Рис. 41 Вибір джерела записів для Форми з наявних таблиць і запитів.

Для створення елемента управління краще використовувати область **Список полів**, оскільки:

–приєднаний елемент має пов'язаний підпис, який приймає ім'я поля за умовчанням (або підпис, визначений для поля в базовій таблиці або базовому запиті), тому не потрібно вводити підпис власноруч;

–приєднаний елемент успадковує багато настройок поля з базової таблиці або базового запиту (зокрема значення властивостей **Формат (Format)**, **кількість знаків після коми (DecimalPlaces)** та **маску вводу (InputMask)**). Тому можна бути впевненим, що ці властивості поля залишатимуться однаковими для кожного елемента, створеного й приєднаного до цього поля.

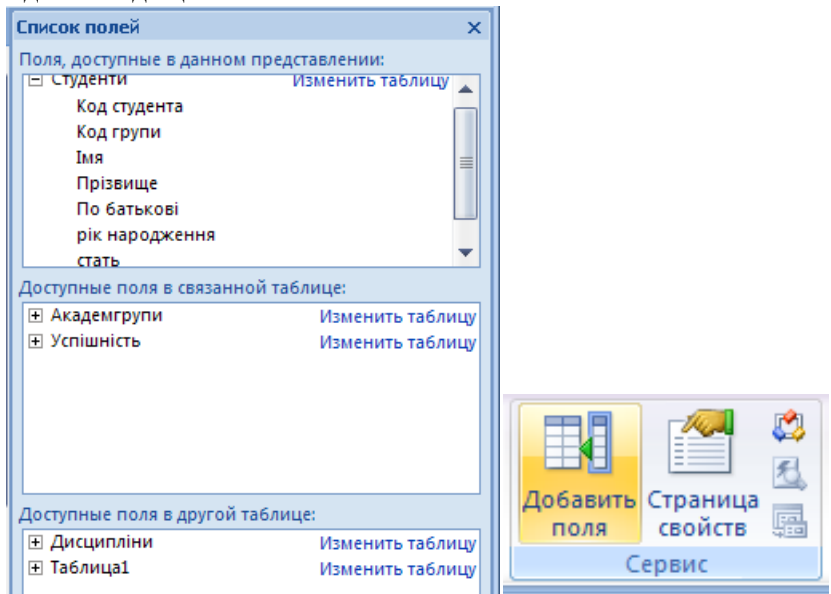



Рис. 42 Вікно Список полів і кнопка, що його відкриває.

Якщо вже створено вільний елемент керування й потрібно приєднати його до поля, в полі властивості **Джерело записів (ControlSource)** для цього елемента керування слід вказати ім'я поля. Для отримання докладніших відомостей про властивість **Джерело записів** виконайте пошук у довідці за ключовим словом «Джерело записів».

Корисною властивістю форми є можливість виконання обчислень над даними полів. Для цього треба ввести, користуючись панеллю елементів, вільне поле, видалити назву Свободный, поставити знак рівності (=) та записати потрібну функцію або будь-який математичний вираз. Як відомо, вирази складаються з змінних, констант, функцій з'єднаних математичними операціями і дужками.

Взагалі як значення властивості поля **Данные** програма передбачає ввід математичних виразів з використанням вбудованих функцій, що входять до комплекту майстра **Построитель выражений** (рис. 42). Він викликається кнопкою  у рядку **Данные** вікна властивостей поля.

У ньому передбачено багато функцій. Як прийнято у всіх мовах програмування, аргументи записуються у дужках. Якщо аргументом є назва поля, вона записується у квадратних дужках. Наприклад:

Sum([вартість]) – визначає суму всіх значень поля **вартість**;

Avg([відстань]) – визначає середнє арифметичне всіх значень поля **відстань**;

Щоб результат обчислень був зрозумілим, треба змінити стандартну назву поля ПолеN на більш змістовну.

Досить часто використовується можливість введення дати та часу. Щоб додати ці дані на формі, треба створити нове вільне поле (наприклад, у ділянці верхнього колонтитулу), відкрити вікно властивостей поля, у рядку **Данные** (вкладка **Данные**) записати =Now() та у рядку **Формат поля** (вкладка **Макет**) задати потрібний формат дати. Дату можна ввести таким же чином за допомогою функції Date().

У виразах текст, що записаний у лапках, відтворюється «буквально», а для «склеювання» тексту, взятого в лапки або значень змінних чи функцій служить оператор &, який називається оператором конкатенації.

Наприклад, для отримання значення віку студентів у формі, що пов'язана з таблицею Студент, створимо нове вільне поле Вік. У властивості **Данные** цього поля введемо вираз, який наведено на рис.43 .

=Year(Date())-[рік народження]

Друге поле, що обчислюється, створимо бінарне поле **Стать жін**. Значення цього поля протилежне значенню поля **Стать чол**, а вираз у властивості **Данные** має вигляд інверсії поля **Стать**: =Not ([стать]).

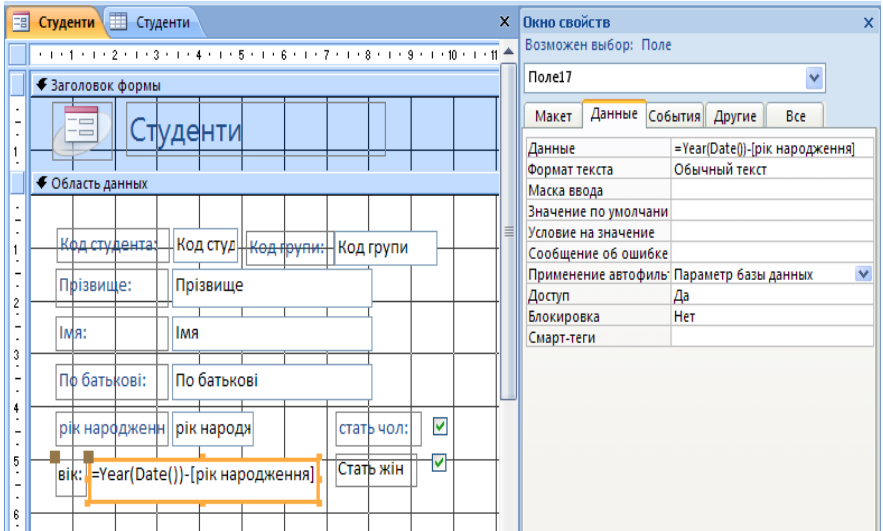


Рис. 43 Поле для обчислень віку у формі Студенти.

Функція Year() вибирає рік з поточної дати, що надає функція Date(). Результати обчислень обох полів наведено на рис. 44. .

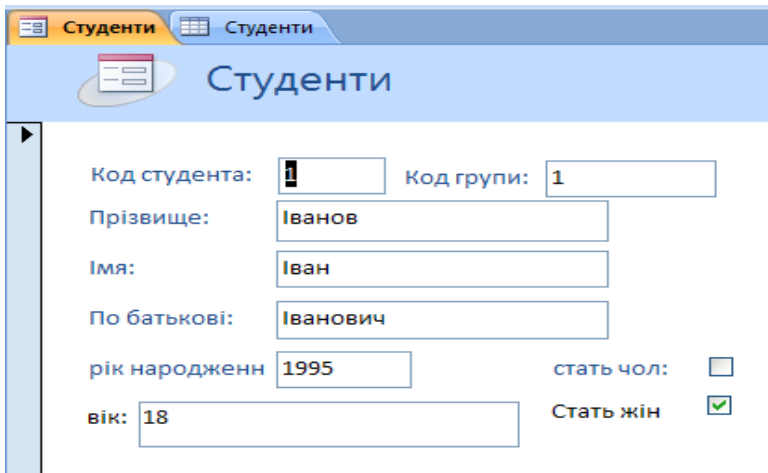


Рис. 44 Поля, що обчислюються (Вік і Стать жінін).

7.5. Складні форми

Для виводу у формі даних з декількох зв'язаних таблиць створюється складна форма, тобто така, коли одна форма (головна) включає до себе іншу (підпорядковану) форму. Підпорядкованих форм може бути декілька. Підпорядкована форма може включати до себе іншу підпорядковану форму. Таким чином, складні форми утворюють ієрархічні структури різних варіантів.

Складна форма поєднує в собі дані із таблиць, між якими встановлений зв'язок один - до - багатьох.

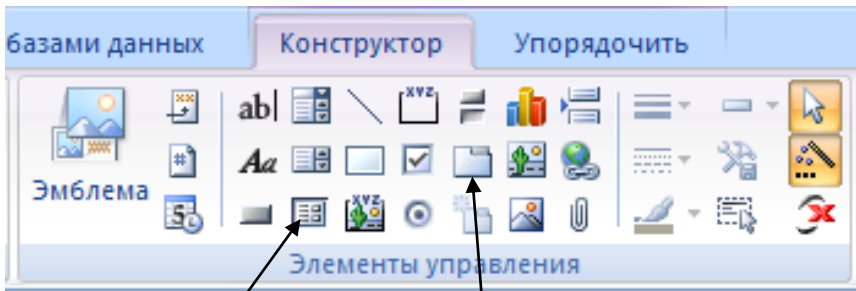
Для створення складної форми також доцільно використовувати автоматичне створення форм, що дозволяє значно прискорити роботу. Підпорядкування форм здійснюється відповідно до існуючого зв'язку між таблицями: таблиця з боку "один" розглядається як головна.

Подальша робота із створеною формою може виконуватися у режимі конструктора, де користувач здійснює корекцію форми відповідно до своїх побажань. Тут повністю справедливі рекомендації, що були сформульовані вище.

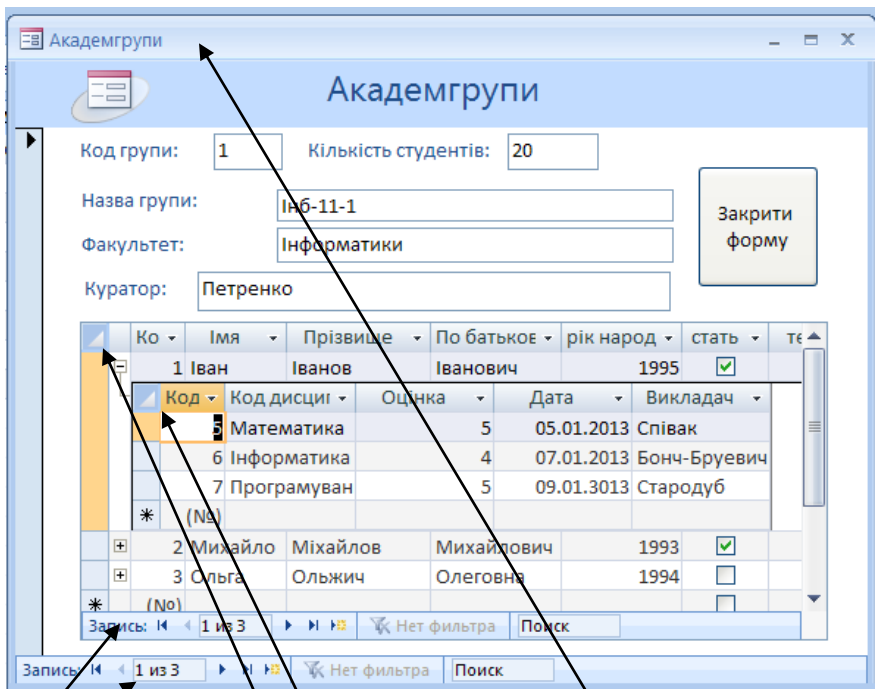
Складна форма дозволяє змінювати дані у таблиці з боку "багатьох" та у неключових полях таблиці з боку "один". Всі зміни будуть враховані у таблицях.

Зв'язати можна вже створені форми, працюючи у режимі конструктора. Спочатку треба відкрити форму, яка вважається головною. Вибрати на панелі елементів інструмент **Кнопка** та розмістити її на вільному місці форми. Відкривається вікно **Создание кнопок**, де надані переліки **Категории** та **Действия**, з яких вибирається варіант **Работа с формой** та **Открытие формы**. У наступних вікнах вибирається форма, яка буде зв'язана, варіант роботи (наприклад, **Открыть форму** и **показать все записи**) та створюється надпис на кнопці. Залишається лише відрегулювати розміри обох форм, коли вони будуть відкриті у режимі таблиці.

Розглянемо створення складної форми з декількома підпорядкованими формами. Якщо не треба виводити всі форми одночасно на екран, може бути прийнятий варіант, коли підпорядковані форми будуть викликатися за необхідністю поодиночі. Цей варіант може бути реалізований з використанням інструментів **Вкладка** та **Подчиненная форма/отчет** в розділі **Конструктор / Елементи управління**.



Підлегла форма/звіт Вкладка
 Рис. 45 Інструменти створення складної форми.



Панелі навігації Друга підлегла форма
 Перша підлегла форма Основна форма

Рис. 46 Складна трьохрівнева форма.

Спочатку створюються всі форми, які будуть відтворюватися, як підпорядковані. Далі створюється (за допомогою майстра або з використанням конструктора) основна форма. Треба встановити її розміри так, щоб було вільне місце для впровадження набору вкладок. Вибравши на панелі елементів інструмент **Вкладка**, курсором (який має вигляд інструмента), починаючи з лівого верхнього кута, накреслюють на формі розміри вікон вкладок. На формі з'являється набір з двома вкладками.

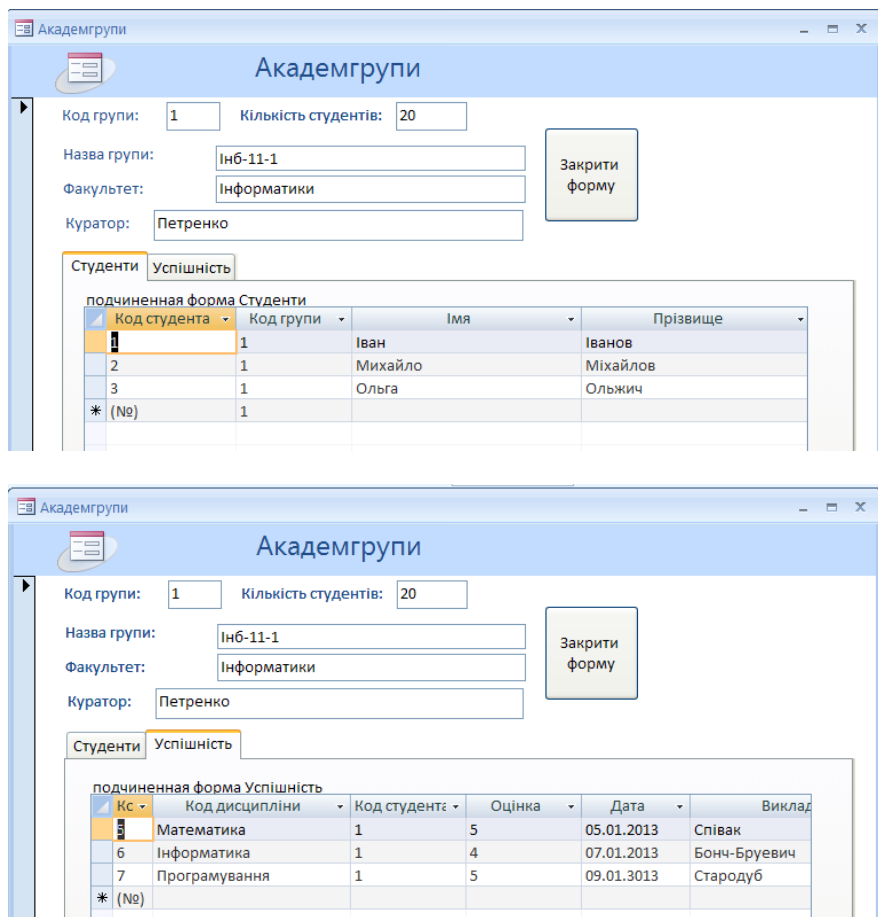


Рис. 47 Розміщення підлеглих форм на вкладках.

Якщо підпорядкованих форм більше, можна додати вкладки, користуючись контекстним меню, яке викликається правим клавішем миші. Далі вибирається інструмент **Подчиненная форма/отчет** та курсором переноситься на першу вкладку у вигляді чорного прямокутника. Кліком лівої клавіші викликається вікно майстра підпорядкованих форм. Тут треба перемикачем вибрати **Формы**, розкрити список та вибрати у ньому форму, яку треба вставити на вкладку. Так, послідовно відкриваючи вкладки, на них заносяться інші підпорядковані форми.

Наприклад, згідно з схемою бази даних (рис. 46) автоматично створена форма **Академгрупи** має три рівня ієрархії (рис. 46). Підлеглі форми, що розміщені на вкладках, зображені на рис. 47.

7.6. Контрольні питання до 7 розділу

1. Що таке форма? Для чого вона використовується?
2. Що є джерелом даних для форми?
3. Якими способами можна створити форму?
4. Як можна додати елемент керування у форму?
5. Де міститься інформація у формі?
6. Чи можна формувати та редагувати елемент керування у формі? Яким чином?
7. Як можна переглядати та доповнювати дані таблиці за допомогою форми?
8. Чи можна створити форму за допомогою якої можна лише переглядати дані таблиць бази даних? Як?
9. В яких режимах можна відобразити форму?
10. Як можна завантажити конструктор форм?
11. Коли варто використовувати конструктор форм?

8. Створення і використання звітів

У СУБД MS Access передбачена можливість друкування таблиць, запитів та форм. Проте як найкращий засіб подання сформованих даних у друкованому вигляді вважаються звіти: саме вони створюються спеціально для виводу даних з урахуванням всіх вимог стосовно розподілу їх на сторінки та розміщення на сторінках.

8.1. Типи звітів

Звіти, що створюються Access розподіляються на такі основні типи, так званих макетів.

Макет у вигляді таблиці. Цій макет нагадує таблицю, де зберігаються дані. Ширина таблиці може перевищувати ширину звіту.

Макет в стовпчик. Кожен фрагмент даних має позначку, поля слідує один за одним в стовпчик. Цей макет добре підходить для звітів, які містять дуже багато полів, щоб їх можна було вивести у форматі таблиці, інакше ширина стовпця перевищить ширину звіту.

Змішаний макет. Допускається змішування табличного макету з макетом в стовпчик. Наприклад, можна розмістити ряд полів по кожній із записів у горизонтальному рядку, а інші поля з того ж запису в стовпчик.

Вирівняний макет. Якщо звіт складається за допомогою майстра звітів, можна скористатися вирівняним макетом. У цьому макеті записи відображаються як можна компактніше з урахуванням повної ширини сторінки. Звичайно, такого результату можна досягти і без використання майстра звітів, проте точне вирівнювання полів - трудомістка процедура.

Вирівняний звіт добре підходить в тому випадку, якщо потрібно відобразити у звіті велика кількість полів. Якщо в попередньому прикладі вивести ті ж дані з використанням макета таблиці, поля зайдуть за край сторінки. При використанні макета в стовпчик для кожного запису буде потрібно набагато більше місця по вертикалі, що призведе до витрати паперу і ускладнить читання звіту.

Макети елементів управління - це нова можливість в Office Access 2007. Макети елементів управління являють собою напрямні, які можна додати до звіту, відкритий в режимі макета або поданні конструктора. У Access 2007 макети елементів управління додаються автоматично, якщо звіт складається за допомогою майстра звітів або для створення звіту була натиснута кнопка Звіт у групі Звіти вкладки Створити. Макет елемента нагадує таблицю, кожна комірка якої містить мітку, текстове поле або будь-який інший тип елемента.

Групи/підсумковий звіти являють собою найбільш розповсюджений тип звітів. В них об'єднуються дані для груп записів, а в кінці звіту вказуються підсумкові значення.

Поштові наліпки являють собою спеціальний тип багатоколонних звітів, що призначені для друку імен та адрес (або інших даних з декількох полів) в групах. Структура паперу для поштових наліпок,

на якому друкуються такі звіти, визначає кількість строк і стовпців на сторінці.

У незв'язаних звітах містяться підлеглі звіти, що засновані на незв'язаних джерелах даних, наприклад, таблицях чи запитах. Основний звіт незв'язаного звіту не використовує у якості джерела таблицю або запит. Але підлеглі звіти, що містяться в незв'язаному, повинні посилатися на джерело даних. Незв'язані звіти дозволяють об'єднувати підлеглі звіти, що зв'язані з незалежними таблицями і запитами.

У звітах інших типів, як і в формах, у якості джерел даних використовуються таблиці або запити. Звіти таких типів називають зв'язаними з джерелами даних.

По типу структури можна відмітити такі звіти:

Простий звіт містить елементи управління, які пов'язані (сполучені) з різними таблиць і запитами. Прикладом такого звіту може служити звіт, який показує студентів і дисципліни.

Ієрархічний звіт містить інші звіти, які називають підлеглими звітами. Підлеглі звіти впроваджуються, коли потрібно відобразити дані з таблиць зі зв'язком «один-до-багатьом». Наприклад, можна зробити звіт «Академгрупи», в який будуть включені дані з таблиць «Студенти» і «Успішність».

Записи в звіті можна сортувати і групувати або в режимі конструктора, або використовуючи майстер форм.

Можна сортувати по одному полю або по декількох полів в джерелі запису і задавати порядок сортування.

Для записів у звіті можна задати один або кілька рівнів угруповання і визначити порядок сортування для кожного рівня. Також можна для кожної групи показувати верхні та нижні колонтитули, підраховувати підсумкові та інші значення.

У якості даних для звіту перехресного типу можна використовувати результати перехресних запитів. У перехресному запиті підраховується сума, середнє значення або інші типи підсумкових значень для даних, а результати групуються за двома полями: значення одного поля стають заголовки стовпців, а значення іншого поля - заголовками рядків.

У звіті з декількох стовпчиків дані відображаються в одному або декількох стовпцях. Кількість стовпчиків, ширину стовпців і відстань між рядками і стовпцями можна налаштувати.

8.2. Створення звітів

Звіт завжди створюється на основі таблиці або запиту (результуюча таблиця). Самий простий спосіб це автоматичне створення звіту, яке багато в чому схоже на процес створення форми. При автоматичному створенні у звіт включаються усі поля відповідної таблиці, які розташовуються в один рядок. Це майже завжди не зручно. Тому звіт корегується у режимі Конструктор.

Створення звітів виконується інструментами меню **Створити (Создание)**, розділ **Звіти (Отчеты)**. Автоматичне створення виконується натисканням на піктограму **Звіт (Отчет)**, а за допомогою майстра натисканням на піктограму **Майстер звітів**. Для уточнення у розміщенні даних, зміни розмірів шрифтів, кольорове оформлення фрагментів звіту можуть бути внесені певні зміни у вигляд звіту.

Під час створення звіту з використанням засобів Звіт або за допомогою майстра звітів в Office Access 2007 автоматично виконується додавання полів у звіт і створення оптимальних елементів управління для відображення цих полів з урахуванням їх типу даних. При самостійному додавання полів у звіт рекомендується перетягнути необхідні поля звіту з області **Список полів**. Для кожного поля буде автоматично створений оптимальний елемент керування на базі типу даних цього поля. Для більшості типів даних полів оптимальним є такий елемент керування, як текстове поле (встановлений за замовчуванням). Редагування звіту в режимі Конструктора дуже подібно до редагування форм в режимі Конструктора. Тут можна здійснити будь-які дії щодо зміни індивідуального вигляду елемента на формі. Якщо зміни стосуються декількох елементів, їх треба виділити, використовуючи, як звичайно, клавішу Shift.

В поданні конструктора можна додати до звіту нові елементи управління та поля. Вікно властивостей надає доступ до великої кількості властивостей, за допомогою яких можна настроїти звіт.

Для переходу до подання конструктора клацніть правою кнопкою миші ім'я звіту в області переходів і виберіть пункт Конструктор. Звіт відображається в поданні конструктора.

Для змінення властивостей звіту, його елементів та розділів можна скористатись властивостями. Щоб відобразити вікно властивостей, натисніть клавішу F4.

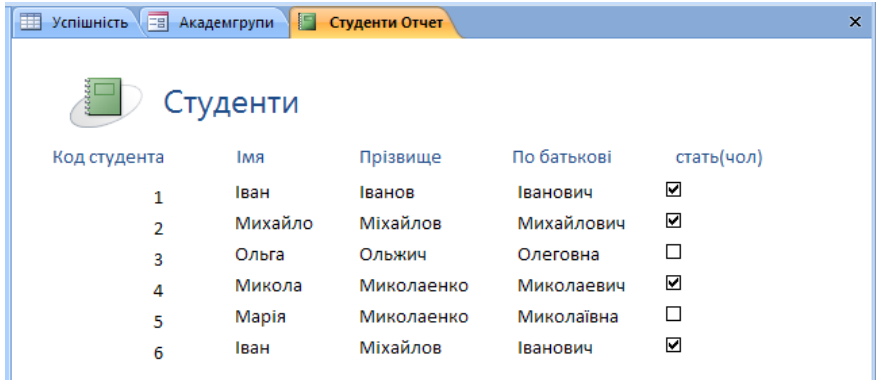
Для додавання полів із базової таблиці або базового запиту до макету звіту можна скористатись областю Список полів. Щоб відобразити область Список полів, виконайте одну з таких дій:

– Виконати команду Конструктор / Сервіс / Додати поля.

–Натисніть сполучення клавіш ALT+F8.

Після цього можна додавати поля, перетягуючи їх з області Список полів до звіту.

Автоматично створений простий звіт зображений на рис. 48 .



The screenshot shows a software window titled "Студенти Отчет" (Students Report). The window has a blue header bar with three tabs: "Успішність" (Success), "Академгрупи" (Academic Groups), and "Студенти Отчет" (Students Report). Below the header, there is a green folder icon and the title "Студенти". The main content is a table with five columns: "Код студента" (Student ID), "Імя" (Name), "Прізвище" (Surname), "По батькові" (Patronymic), and "стать(чол)" (Gender/Male). The table contains six rows of student data.

Код студента	Імя	Прізвище	По батькові	стать(чол)
1	Іван	Іванов	Іванович	<input checked="" type="checkbox"/>
2	Михайло	Міхайлов	Міхайлович	<input checked="" type="checkbox"/>
3	Ольга	Ольжич	Олеговна	<input type="checkbox"/>
4	Микола	Миколаенко	Миколаевич	<input checked="" type="checkbox"/>
5	Марія	Миколаенко	Миколаївна	<input type="checkbox"/>
6	Іван	Міхайлов	Іванович	<input checked="" type="checkbox"/>

Рис. 48 Приклад простого звіту.

8.3. Структура звітів

Як і форми, звіти складаються з розділів, а розділи можуть містити елементи управління. Але, на відміну від форм, розділів у звітах більше, а елементів керування, навпаки, менше. З структурою звіту простіше усього ознайомитися відкривши його в режимі Конструктора. Структура звіту складається з п'ятих основних розділів: заголовка звіту, верхнього колонтитула, області даних, нижнього колонтитула і примітки звіту. У порівнянні з формами новими є розділи верхнього і нижнього колонтитулів (рис. 49).

Переглянути ці розділи можна, відкривши звіт у поданні конструктора. Для створення повноцінних звітів необхідно розуміти засади роботи кожного розділу. Наприклад, спосіб обчислення результату залежить від розділу, в якому розташований обчислюваний елемент. У нижченаведеному списку подано зведені відомості про розширені типи розділів і їх використання:

Заголовок звіту. Цей розділ друкується лише один раз на початку звіту. Зазвичай використовується для даних, які відображаються на титульній сторінці, таких як емблема, заголовок або дата. Якщо тут розташувати обчислюваний елемент керування, в якому використову-

ється агрегатна функція Sum, сума обчислюватиметься для всього звіту. Заголовок звіту друкується перед верхнім колонтитулом сторінки.

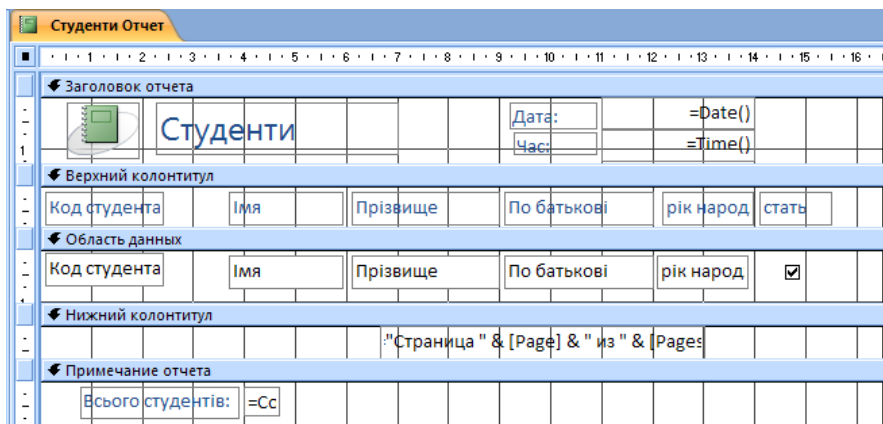


Рис. 49 Звіт у режимі Конструктор.

Верхній колонтитул сторінки. Цей розділ друкується у верхній частині кожної сторінки. Наприклад, він використовується для повторення заголовку звіту на кожній сторінці.

Верхній колонтитул групи. Цей розділ друкується на початку кожної нової групи записів. Він використовується для друку назви групи. Наприклад у звіті, згрупованому за товарами, верхній колонтитул групи використовується для друку назви товару. Якщо у верхньому колонтитулі групи розташувати обчислюваний елемент, в якому використовується агрегатна функція Sum, сума обчислюватиметься для поточної групи.

Область даних. Цей розділ друкується один раз на кожен рядок джерела записів. У ньому розташовуються елементи керування, які складають основний вміст звіту.

Нижній колонтитул групи. Цей розділ друкується наприкінці кожної групи записів. Він використовується для друку підсумкової інформації за групою.

Нижній колонтитул сторінки. Цей розділ друкується наприкінці кожної сторінки. Він використовується для друку номерів сторінок або даних, які стосуються кожної сторінки.

Нижній колонтитул звіту. Цей розділ друкується лише один раз наприкінці звіту. Він використовується для друку підсумків або іншої підсумкової інформації за всім звітом.

Примітка. Розділ примітки використовують для розміщення додаткової інформації.

Пустий звіт має три розділи: Верхній колонтитул, Область даних, Нижній колонтитул.

У поданні конструктора нижній колонтитул звіту розташований під нижнім колонтитулом сторінки. Але під час друку або попереднього перегляду нижній колонтитул звіту буде розташований над нижнім колонтитулом сторінки, відразу після нижнього колонтитула останньої групи або після рядка даних на останній сторінці.

Розділ заголовка служить для друку загального заголовка звіту. Розділ верхнього і нижнього колонтитула можна використовувати для друку підзаголовків, якщо звіт має складну структуру і займає багато сторінок. В нижньому колонтитулі можна також поміщати номери сторінок.

У області даних розміщують елементи управління, пов'язані з вмістом полів таблиць бази. Записи в області даних повторюються стільки раз, скільки їх у таблиці. Порядок розміщення і вирівнювання елементів керування той же, що і при створенні структури форм.

Корисною властивістю звіту є можливість групувати дані за допомогою вікна **Сортировка и группировка**. У вікні можна визначити до 10 полів, які будуть використані у звіті для розподілу даних на групи. Поле у першому рядку списку визначає основну групу, подальші поля – підгрупи, що вкладаються одна у іншу. Робота у цьому вікні визначає структуру звіту. За замовчуванням заголовки та примітки груп отримують назву з використанням імені поля або вирази, на якому заснована група. Після цього залишається лише у відповідних ділянках розмістити потрібні поля. На рис. 50 зображено вікно конструктора для створення звіту з групуванням за рік народження студентів, а на рис. 51 – цей звіт у режимі подання звіту.

Для додавання колонтитулів сторінки або заголовка та примітки звіту в області переходів клацніть правою кнопкою миші звіт, який потрібно змінити, і виберіть у контекстному меню пункт Конструктор.

Щоб додати у звіт розділи верхнього або нижнього колонтитула або заголовка або примітки звіту, клацніть правою кнопкою миші будь-яке з областей виділення розділів і виберіть у контекстному меню команду **Колонтитули** або **Заголовок/примітка звіту**.

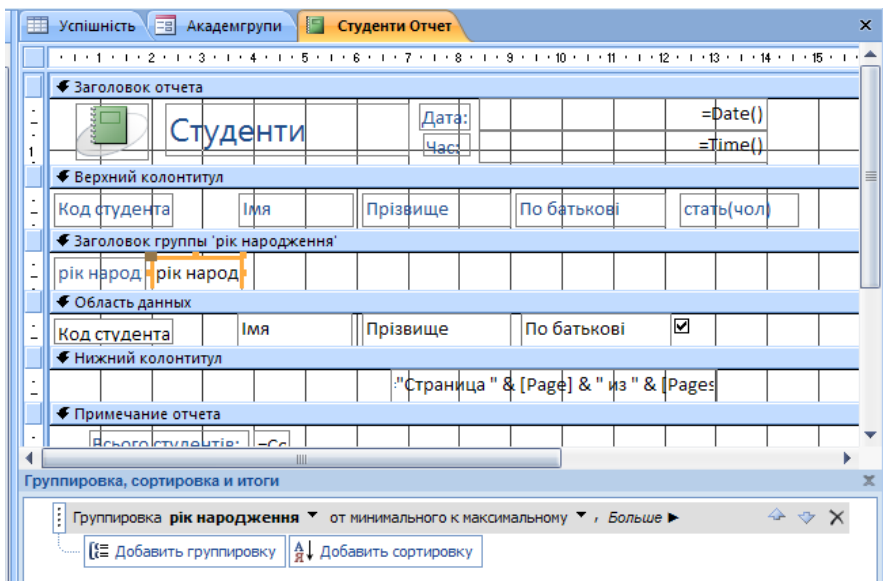


Рис. 50 Звіт з групуванням за роком народження.

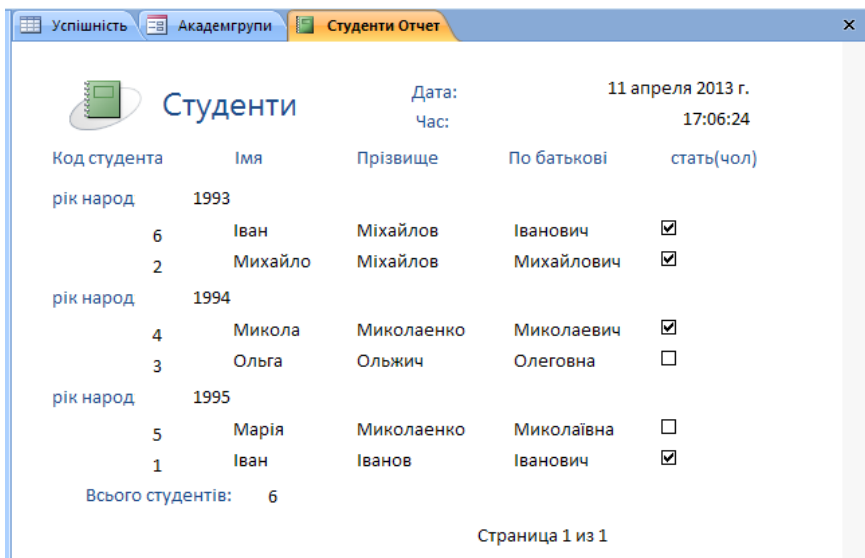


Рис. 51 Видяг звіту з групуванням.

Після цього можна перемістити в нові розділи існуючі елементи управління або додати нові.

В Office Access 2007 заголовок і примітка звіту або верхній і нижній колонтитул сторінки завжди додаються разом. Іншими словами, не можна додати розділ заголовка звіту або верхнього колонтитула сторінки без примітки звіту або нижнього колонтитула сторінки відповідно. Якщо один з таких розділів не потрібен, його можна видалити, проте можна змінити розмір незатребуваного розділу, встановивши для нього висоту нуль (0), щоб не витрачати місце в звіті. Встановіть вказівник миші в нижній частині невикористаного розділу, щоб покажчик прийняв вид двосторонню стрілку, а потім потягніть вгору, щоб приховати розділ. При наявності в розділі елементів управління необхідно попередньо видалити їх, щоб повністю приховати розділ.

Для видалення колонтитулів сторінки або заголовка та примітки звіту в області переходів клацніть правою кнопкою миші звіт, який потрібно змінити, і виберіть у контекстному меню пункт Конструктор.

Клацніть правою кнопкою миші область виділення розділу і виберіть у контекстному меню команду Колонтитули або Заголовок/примітка звіту.

При видаленні пари розділів, у якій містяться елементи управління, в Office Access 2007 виводиться попередження про те, що видалення даних розділів призведе до видалення елементів керування без можливості скасування дії. Натисніть кнопку Так, щоб видалити розділ разом з елементами управління, або кнопку Ні, щоб скасувати дію.

У звітах також використовують елементи управління, які відображають дані, виконують дії та надають можливості перегляду й роботи з інформацією, що вдосконалює інтерфейс користувача (підписами, зображеннями тощо). У звітах як і у формах використовуються три типи полів: приєднані, вільні та обчислювані.

8.4. Обчислення у звітах


Використання обчислюваного елемента управління у звітах співпадає з його використанням у формах. Це елемент управління, джерелом даних якого є вираз, а не поле. Значення, яке слід використовувати як джерело даних для цього елемента, вказується через визначення виразу.

Створення виразів розглянуто у розділі 6. Вираз може використовувати дані з поля базової таблиці чи запиту або з елемента управління у звіті.

Елемент управління приєднується до поля через ідентифікацію поля, з якого елемент отримує дані. Можна створити елемент, приєднаний до вибраного поля, перетягнувши поле з області Список полів до звіту. В області Список полів відображаються поля базової таблиці або базового запиту звіту. У цьому випадку автоматично створюється підпис і успадковуються властивості поля з базової таблиці.

Інший спосіб приєднання поля до елемента керування — ввести ім'я поля в самому елементі керування або в полі значення Джерело записів (ControlSource) у вікні властивостей елемента керування.

Якщо вже створено вільний елемент й потрібно приєднати його до поля, в полі властивості Джерело записів (ControlSource) для цього елемента керування слід указати ім'я поля.

Корисною властивістю звіту є можливість виконання обчислень над даними полів. Взагалі як значення властивості Данние програма передбачає ввід математичних виразів з використанням вбудованих функцій, що входять до комплексу майстра **Построитель выражений**. Он викликається кнопкою .

Обчислення здійснюються також при визначенні підсумків за даними полів (статистична обробка даних у полях). Для цього треба у ділянці приміток групи, що обробляється, ввести, користуючись панеллю елементів, вільне поле, видалити назву Свободный, поставити знак рівності (=) та записати потрібну функцію або будь-який математичний вираз.

У програмі передбачено вісім функцій (рис 52): Sum, Avg, Max, Min, Count, Var, StDev, StDevP. Як прийнято у всіх мовах програмування, аргументи записуються у дужках. Якщо аргументом є назва поля, вона записується у квадратних дужках. Приклади запису функцій:

Sum([вартість]) – визначає суму всіх значень поля **вартість**;

Avg([відстань]) – визначає середнє арифметичне всіх значень поля **відстань**;

Count([прізвище]) – визначає число записів поля **прізвище**.

Функції можуть записуватися для виділеного поля також у рядку **Данние** на вкладці **Данние** у вікні **Вікно властивостей**. При цьому можна використовувати майстра **Построитель выражений**, що певною мірою спрощує роботу.

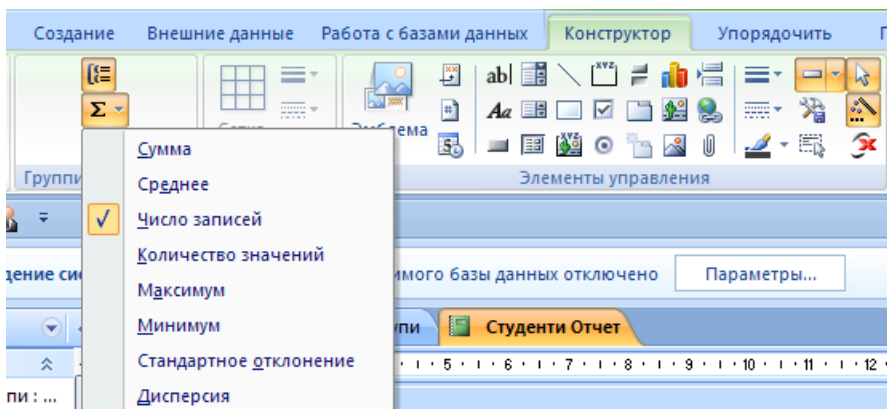


Рис. 52 Функції, що додаються кнопкою Підсумки (Итоги).

Щоб підсумок був зрозумілим, треба змінити стандартну назву поля ПолеN на більш змістовну.

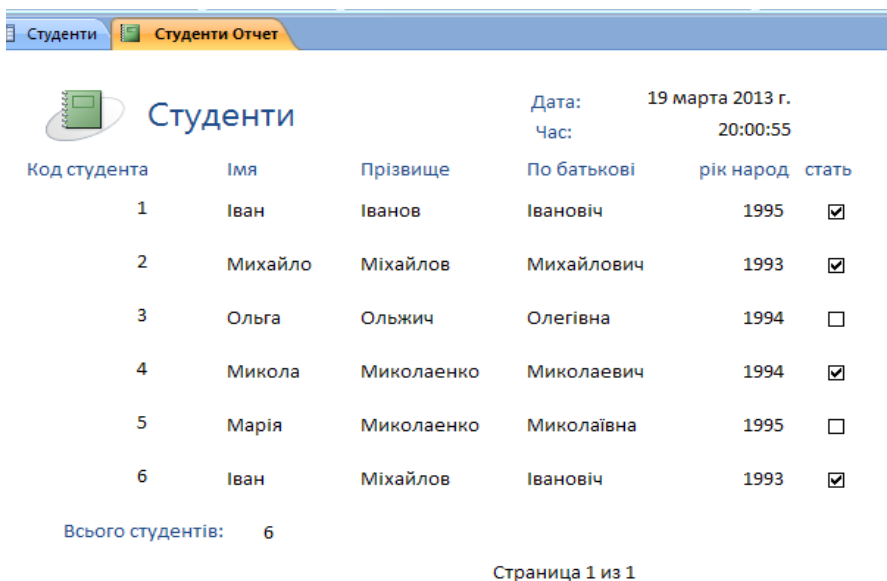


Рис.53. Приклад звіту з полями, що обчислюються.

Під час створення звіту передбачена можливість введення дати та часу. Щоб додати ці дані у звіт, треба створити нове вільне поле

(наприклад, у ділянці верхнього колонтитулу), відкрити вікно властивостей поля, у рядку Данные (вкладка Данные) записати =Now() та у рядку Формат поля (вкладка Макет) задати потрібний формат дати. Дату можна ввести таким же чином за допомогою функції Date(). Вона повертає поточну дату і поміщає її в поле, а звіт відтворює її при друці.

В нижньому колонтитулі є елементи керування для виводу номера сторінки і загальної кількості сторінок. Для їхнього визначення використані вмонтовані функції Page() і Pages(). Той текст, що записаний у лапках, відтворюється «буквально», а оператор служить для «склеювання» тексту, взятого в лапки, із значеннями, що повертаються функціями. Оператор & називається оператором конкатенації.

На рис. 53 показаний приклад звіту з обчислюваними полями.

8.5. Перегляд звіту

Існує чотири режиму переглядання звіту. Вибір режиму залежить від дій, які планується здійснити зі звітом чи його даними. Ці режими: Подання звіту, Попередній перегляд, режим Макета і Конструктор. Вибір режиму здійснюється кнопкою **Главная / Режимы / ...**

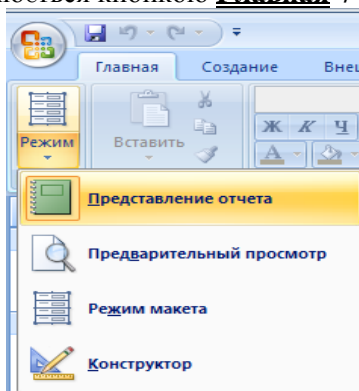


Рис. 54 Вибір режиму перегляду звіту.

Подання звіту використовується коли потрібно внести тимчасові зміни до відображених даних у звіті перед здійсненням друку, або якщо потрібно скопіювати дані до буфера обміну.

Режимом макета використовується якщо потрібно мати змогу змінити структуру звіту під час перегляду даних.

Режимом попереднього перегляду якщо потрібно просто переглянути, як виглядатиме звіт після друку.

Подання звіту використовується за умовчанням у разі відкриття звіту подвійним клацанням в області переходів. Якщо звіт не відкрито, двічі клацніть його в області переходів, щоб відкрити його в режимі звіту.

У режимі Конструктор здійснюється створення звітів, змінення і налаштування його вигляду.

Якщо звіт уже відкрито, клацніть правою кнопкою миші ім'я звіту в області переходів і виберіть пункт Подання звіту. Перехід від одного режиму до другого також здійснюється через контекстне меню.

8.6. Контрольні питання до 8 розділу

1. Що таке звіт? Для чого він використовується?
2. На основі яких об'єктів бази даних можна створити звіт?
3. Якими способами можна створити звіт?
4. В яких режимах можна відобразити звіт?
5. Опишіть процес створення звіту за допомогою майстра.
6. Як можна додати елемент управління у звіт?
7. Чи можна форматувати та редагувати елемент керування у звіті? Яким чином?
8. Для чого використовується кожна область проекту звіту?
9. Чи можна редагувати області звіту? Яким чином?
10. Для чого використовується групування при створенні звіту?

9. Створення макросів

9.1. Поняття макросу

Для виконання одних і тих самих нескладних задач у системі Access 2007 передбачені макроси. Функціонально можливостями вони поступаються мові VBA, але макроси створюються значно простіше і швидше.

Макрос — це одна або сукупність кількох макрокоманд, тобто інструкцій, за якими виконується певна послідовність дій. Усього в системі Access 2007 є більше 70 макрокоманд, які можна об'єднувати у функціональні групи, наприклад, макрокоманди роботи з об'єк-

тами БД, макрокоманди імпорту й експорту, макрокоманди керування процесами тощо. В системі Access 2007 можна працювати не тільки на рівні окремих макросів, а й об'єднувати макроси в групи із загальним іменем. Таке об'єднання дозволяє виконувати одразу кілька задач.

Виконання макросів прив'язується до певних подій, які трапляються з об'єктами, формами і елементами управління. Усі вони мають події, що відображаються у вікні властивостей і там же указуються макроси або процедури VBA, які виконуються по цим подіям.

9.2. Створення макросів

Для створення макросу відкривається вкладка **Створити (Создание)** і в групі **Інші (Другие)** виконується команда **Макрос/Макрос**. З'являється вікно для створення макросів (рис. 55).

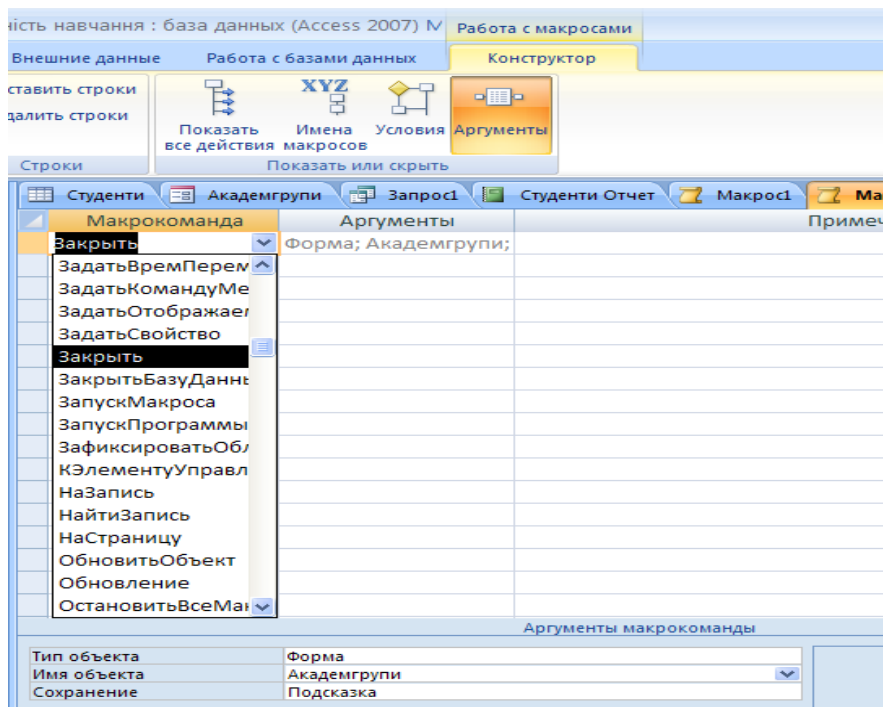


Рис. 55 Вікно створення макросів з відкритим списком макрокоманд.

У стовпці **Макрокоманда** цього вікна можна відкрити список макрокоманд. **Макрокоманда** — це операція, інструкція, яка виконується за макрокомандою, наприклад, відкрити таблицю, надрукувати звіт, висвітити на екрані певне повідомлення тощо. **Аргументи** — це параметри макрокоманди. В кожній макрокоманді є свої аргументи. Наприклад, в макрокоманді **Відкрити таблицю** є такі три аргументи: **Ім'я таблиці** (одна з таблиць, зареєстрованих у поточній БД), режим **Подання** (доступні такі режими: Таблиця, Конструктор, Попередній перегляд, Зведена таблиця, Зведена діаграма) і **Режим даних** (Додати, Редагувати, Лише для читання). У стовпці **Примітка** можна записувати будь-який пояснювальний текст до макрокоманди.

За необхідності можна додати колонки: **Ім'я макросу** дозволяє створювати групу макросів, кожний з яких має власне ім'я, а в **Умови** можна вказати значення, за яким буде виконуватися або не виконуватися макрокоманда, записана у цьому самому рядку. З'являються ці колонки при натисканні на піктограму **Імена макросов** або **Умови** у групі **Конструктор** / **Показати или скрыть**.

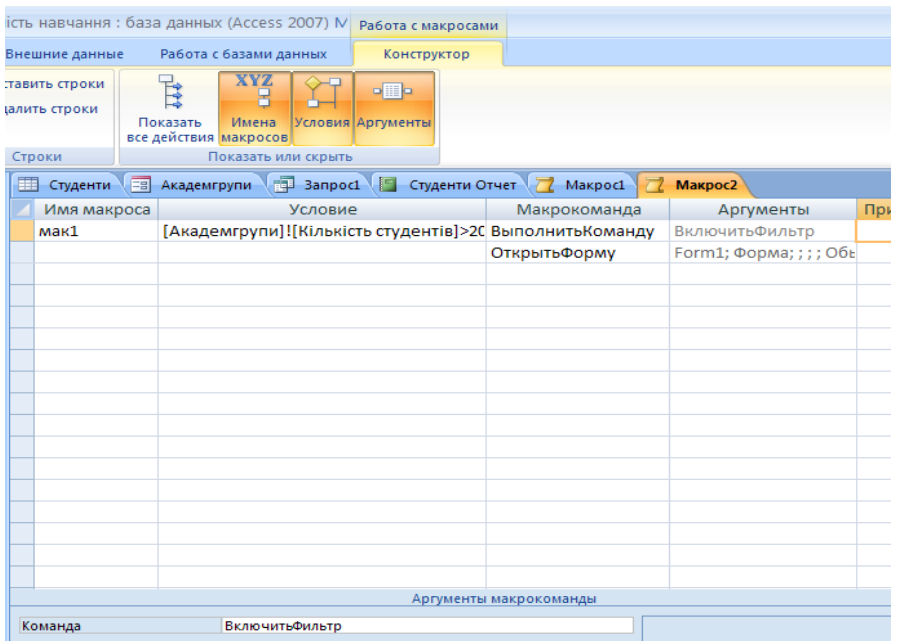


Рис.56 Конструктор макросів з додатковими полями.

В деяких макрокомандах аргументів немає. Але в більшості макрокоманд є від одного до шести аргументів. Конкретні назви аргументів і їх допустимі значення закріплені за кожною макрокомандою. Значення аргументів встановлюються за замовчуванням і можуть змінюватися користувачем. Нижче наведені назви деяких макрокоманд та їх призначення.

Дія	Призначення
Виконати макрос	Запуск макросу з іншого макросу, повтору тощо
Відкрити запит	Відкрити запит у режимі попереднього перегляду, таблиці, конструктора тощо з метою редагування, тільки для читання та ів.
Вийти	Завершення роботи в MS Access
Закрити	Закрити вікно вказаного об'єкта БД
Зупинити макрос	Припинення виконання поточного макросу
Перейти до запису	Перейти до вказаного запису в таблиці, формі
Повідомлення	Відображення вікна з вказаним повідомленням
Покроково	Призупинення виконання макросу та перехід до діалогового вікна «Покрокове виконання макросу»
Сигнал	Подання звукового сигналу

Зазначимо, що за допомогою макросів можна розширити й додатково налаштувати групи команд, панель інструментів і меню, пов'язати з кнопкою виконання необхідних функцій. Макроси доцільно створювати для передавання даних між таблицями, відкривання кількох форм та звітів, для контролю даних, що вводяться, під час заповнення форм тощо.

Уводити макрокоманди можна різними способами, у тому числі шляхом введення з клавіатури. Але зручніше використовувати метод вибору команд і аргументів у режимі **Конструктор**, тому що у цьому випадку ймовірність помилок введення зменшується.

Можна створювати макрос із покроковим виконанням макрокоманд. У цьому випадку можна спостерігати результат виконання кожної окремої команди. У першому рядку стовпця відкриваємо список макрокоманд, знаходимо макрокоманду **Покроково** і натискаємо на її імені кнопку миші. У результаті ця макрокоманда буде записана в першому рядку. У цій макрокоманді немає аргументів.

У другому рядку вводимо будь-яку макрокоманду (наприклад, **Перейти до запису**), а в області Аргументи з'являються назви аргу-

ментів і рядки для введення їх значень у нижній частині вікна на панелі **Аргументи макрокоманди**. Після введення аргументів вони одразу з'являються у стовпці **Аргументи** рядку макрокоманди. Для кожного аргументу праворуч на панелі аргументів висвічується його детальне пояснення.

Аналогічно вводимо наступні макрокоманди та їх аргументи. У кінці можна додати команду **Повідомлення** з аргументом «Кінець» і команду **Закрити**. Для збереження макросу на панелі швидкого доступу натискаємо кнопку **Зберегти**. З'явиться вікно, у яке вводимо ім'я і натискаємо кнопку ОК. Ім'я макросу з'явиться в області переходів. Закриваємо макрос.

Виконання макросу можна зробити різними способами. Найпростіший з них — встановити курсор на його імені в області переходів і двічі натиснути кнопку миші. При команді **Покрокове** відкривається вікно **Покрокове виконання макросу**. Для виконання першої макрокоманди натискаємо кнопку **Крок**. Потім аналогічно виконуємо другу макрокоманду і спостерігаємо результати.

На третьому кроці з'явиться вікно з текстом **Кінець**, у якому натискаємо кнопку ОК. На четвертому кроці закриється макрос. Якщо макрос виконується неправильно, необхідно знайти помилку, ліквідувати її і запустити його повторно.

Створений макрос можна редагувати у режимі **Конструктора**, зокрема, можна змінювати макрокоманди та їх аргументи, додавати і вставляти нові макрокоманди, вилучати і копіювати їх. Для відкривання макросу у режимі **Конструктора** необхідно відкрити його контекстне меню і виконати команду **Конструктор**.

Макроси можна вилучати, перейменовувати і створювати нові копії. Для цього необхідно відкрити його контекстне меню і виконати необхідну команду (Видалити, Перейменувати або Копіювати).

9.3. Макроси з умовними макрокомандами

Макрокоманди можна виконувати одну або кілька підряд записаних залежно від певної умови. Якщо умова не виконується, то макрокоманда (серія макрокоманд) також не виконується. Умова задається логічним виразом, який може набувати двох значень: істинно або хибно. У логічному виразі можуть використовуватися логічні операції (AND, OR, NOT), математичні операції (+, /, -, *), константи, функції, імена полів об'єктів БД, елементи керування та їх параметри.

Логічний вираз записується у стовпець Умови (**Условия**), який створюється командою **Создание / Другие / Макрос**. Імена об'єктів БД та їх полів записуються у квадратних дужках. Якщо в умові вказується об'єкт БД та його поле, то між квадратними дужками вказується знак оклику. Наприклад, вираз:

`[Студенти]![вік]>=25`

буде мати значення істинно, якщо в поточному запису таблиці Студенти вік буде не менше 25.

Перед виконанням макрокоманди обчислюється значення записаного у ній логічного виразу. Якщо він має значення істинно, то виконується ця команда й усі підряд записані макрокоманди, у стовпці **Умова** яких знаходяться три крапки (...). Після цього виконується макрокоманда, стовпці **Умова** яких порожні. Якщо ж логічний вираз має значення хибно, то ця команда й усі ті підряд записані макрокоманди, у стовпці **Умова** яких записані три крапки (...), пропускаються, а виконується перша команда, стовпець **Умова** яких порожній, або в ньому записана інша умова.

9.4. Макроси у формах

Інколи макроси доцільно вбудовувати в об'єкти бази даних: форми та звіти. Інтегровані макроси запускаються автоматично у процесі роботи з цими об'єктами у випадках, коли виникає певна подія.

Подія — це умова, яка може виникати у процесі роботи з формою і яка призводить до запуску макросу. Кількість і назви таких подій можуть бути різними. Це залежить від тієї області форми, з якою здійснюється робота в даний час. Для виклику на екран переліку подій для певної області форми необхідно у відкритій формі в режимі конструктора або режимі форми встановити курсор миші на цю область і натиснути праву кнопку миші. У меню, що відкривається, виконати команду **Властивості**. На екрані з'явиться Вікно властивостей, у якому слід відкрити вкладку **Подія**. У цьому вікні відкриється повний перелік подій для обраного об'єкту (рис. 57).

Тут наведена значна кількість подій. Наприклад, подія **Внесені зміни** виникає тоді, коли вносяться перші зміни у форму. Подія **Змінення** виникає кожного разу, коли вносяться будь-які зміни у форму, наприклад, змінюється значення певного поля. Для обробки кожної події призначається процедура VBA або один зі створених раніше макросів.

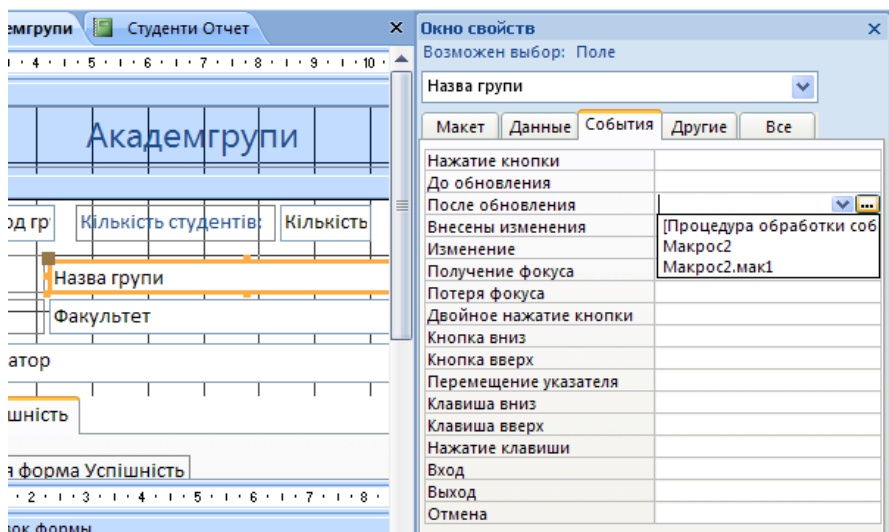


Рис. 57 Перелік подій для поля з ім'ям **Назва групи**.

Нагадаємо, що кілька макросів можуть об'єднуватися у групу із загальним іменем. Зазвичай, у групу об'єднуються макроси, створені для одного й того самого об'єкта БД, або макроси, функції яких деякою мірою співпадають. Правила створення групи макросів принципово не відрізняється від правил створення одиночного макросу. Ім'я групи макросів заноситься в область переходів, а імена окремих макросів вказуються у стовпці **Ім'я макросу** вікна конструктора макросів.

Макроси у групі відокремлюються один від одного порожнім рядком. Під час запуску макросу, який входить до складу групи, спочатку вказується ім'я групи, а потім через крапку ім'я певного макросу.

Дуже часто макроси виконуються при натисканні на кнопки. Розглянемо створення кнопки, яка запускає певний макрос. Відкриваймо будь-яку форму у режимі конструктора. Натискаємо кнопку **Конструктор** / **Елементи управління** / **Использовать мастера** і розміщаємо на формі елемент управління **Кнопка** (рис. 58). Автоматично з'являється вікно **Создание кнопок** (рис. 59). Виберемо, наприклад, в категорії **Работа с формой** пункт **Закрити форму**, а інші опції по умовчужанню. Отримаємо кнопку **Закрити форму**.

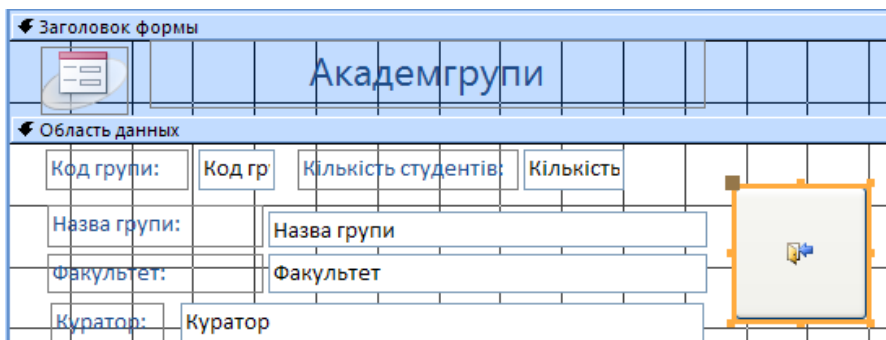


Рис. 58 Розташування кнопки на формі.

У вікні **Вікно свойств** (рис. 60) можна змінити властивості кнопки. Наприклад, властивість **Підпис** можна змінити на «Закрити форму», яка з'явиться кнопці. Варіант кнопки з підписом зображений на рис. 61.

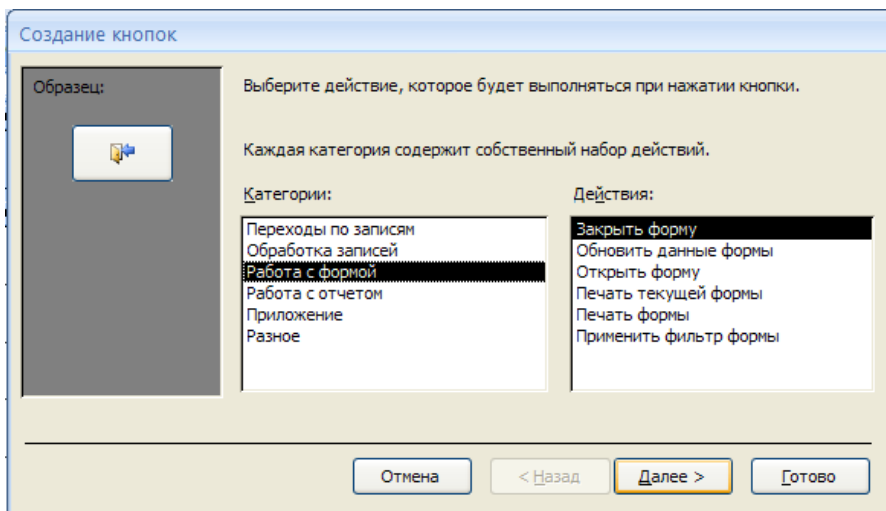


Рис. 59 Вікно для зв'язування макросів з кнопкою.

За допомогою макросу можна змінити будь-які властивості елементів управління. Наприклад змінимо колір фону поля Прізвище форми Студенти при наведенні курсору на це поле. У властивостях по-

ля **Перемещение указателя** натиснемо кнопку і у вікні **Построитель** треба указати Макрос (рис. 62).

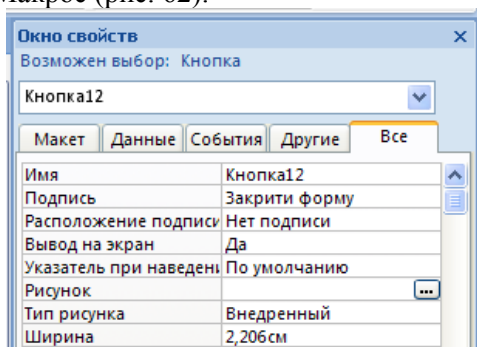


Рис. 60 Влаивности кнопки.

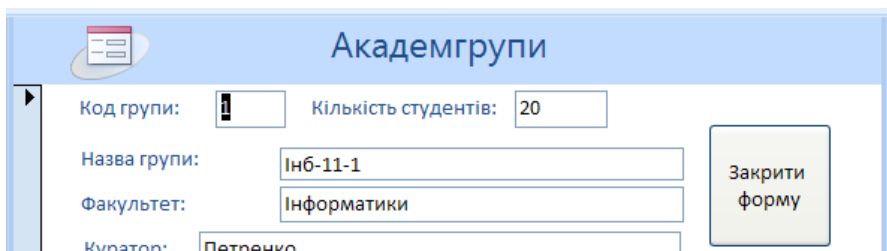


Рис. 61 Новий підпис кнопки.

Відкривається вікно створення макросів (рис. 63). Для прикладу макрос має аргумент **Цвет фона** що дорівнює 65000. Це відповідає жовтому кольору, аргумент 65200 це зелений.

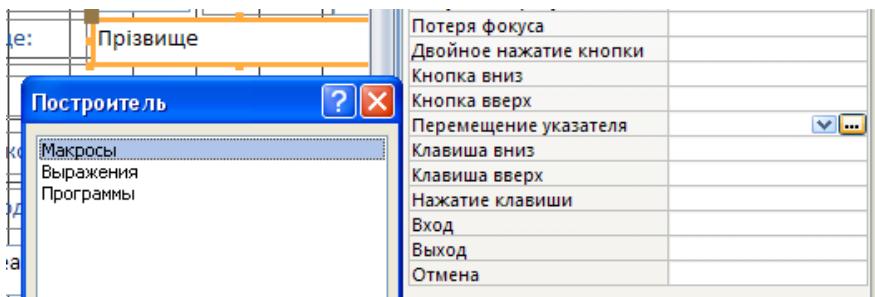


Рис. 62 Призначення події поля виконання макросу, що змінює фон поля.

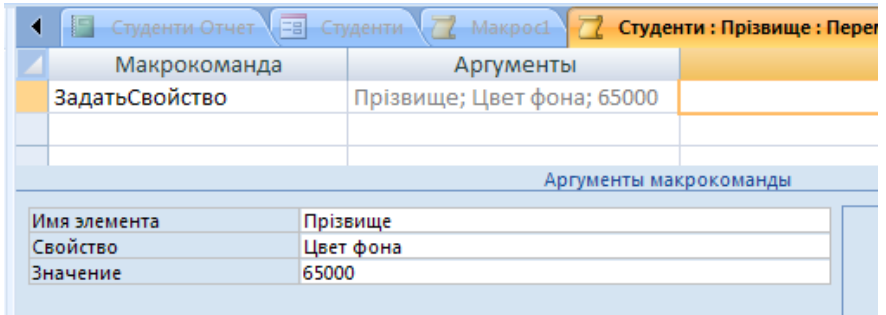


Рис. 63 Макрос, що змінює фон поля Прізвище.

9.5. Контрольні питання до 9 розділу

1. Що таке макрос? Для чого він використовується?
2. Яка структура макросів?
3. Як додаються умови виконання макросу?
4. Які бувають аргументи макросу?
5. Що таке групи макросів?
6. Як макрос прив'язується до об'єктів БД?
7. Що таке подія?
8. Як створити макрос?

10. Додатки

10.1. Додаток 1. Створення шаблону пошуку.

Даний набір підстановочних символів застосовується при використанні діалогового вікна Пошук і заміна з метою пошуку та заміни (якщо потрібно) даних в базі даних Access. Ці знаки можна використовувати в умовах запитів на вибірку і запитів на оновлення бази даних Access.

Знак	Описання	Приклад
*	Відповідаєлюбій кількості знаків. Зірочку (*) можна використовувати в будь-якому місці текстової строки.	чт* — пошук слів «что», «чтобы» та «чтение», але не «почта» або «чат».
?	Відповідає одному із знаків алфавіту.	д?м — пошук слів «дом», «дым» та «дам».

[]	Відповідає одному із знаків в дужках.	д[!оы]м — пошук слів «дом» та «дым», але не «дам».
!	Відповідає одному із знаків, крім знаків в дужках.	д[!оы]м — пошук слова «дам», але не «дом» або «дым».
-	Відповідає любому знаку з діапазону. Необхідно вказувати цей діапазон по збільшенню (от А до Я, але не від Я до А).	б[а-в]г — пошук поднання «баг», «ббг» та «бвг».
#	Відповідає любій цифрі.	1#3 — пошук значень 103, 113, 123
^	Відповідає любому знаку, крім знаків в дужках.	д[^оы]м — пошук слова «дам», але не «дом» або «дым».
≠	одна довільна фраза.	

Щоб знайти символи підстановки, що знаходиться в даних, його слід укласти в квадратні дужки, наприклад так: [#]. Цим правилом повинні дотримуватися при пошуку зірочок (*), знак питання (?), знаків решітки (#), квадратні дужки що відкриваються ([) і дефісів (-). При пошуку знаків оклику (!) або квадратних дужок що закриваються (]) не потрібно укладати їх у дужки. Щоб знайти ці знаки з допомогою діалогового вікна **Пошук і заміна**, введіть знак у полі **Зразок пошуку**, не використовуючи квадратні дужки. За таким же принципом виконується пошук знаків з допомогою запиту. Наприклад, наступний синтаксис повертає всі записи, що містять знак оклику, незалежно від позиції знака в даних: Like "*!*".

При пошуку дефіса разом з іншими знаками слід розмістити дефіс до або після всіх інших знаків в дужках, наприклад: [-#*] або [#*-]. Однак, якщо після початкової дужки розташовується знак оклику (!), дефіс слід помістити після знака оклику: [!-].

При пошуку пари квадратних дужках (відкриває і закриває, []) обидва знаку слід укласти в квадратні дужки: [[]]. Це необхідно, оскільки Access обробляє одну пару як порожній рядок.

У таблиці нижче наведено типи даних, для яких можна виконувати пошук з допомогою символів знаків. Слід пам'ятати, що в деяких

випадках знаки підстановки можна використовувати тільки в діалоговому вікні **Пошук і заміна**, але не в запитах, і навпаки.

Тип даних	Де використовуються підстановочні знаки
------------------	--

«Текстовый»	Діалогове вікно. Пошук та заміна, запити
«Поле MEMO»	Діалогове вікно. Пошук та заміна, запити
«Числовой»	Діалогове вікно. Пошук та заміна, запити
«Дата/Время»	Діалогове вікно. Пошук та заміна, запити

«Денежный»	Діалогове вікно Пошук та заміна, запити
«Счетчик»	Діалогове вікно. Пошук та заміна, запити.
«Объект OLE»	Не використовується.

«Да/Нет» Запити, але в них немає необхідності. Додаткові відомості див. в примітках в кінці розділу

«Гиперссылка»	Діалогове вікно. Пошук та заміна, запити.
---------------	---

«Создание подстановки» В залежності від типу даних вихідного поля.

Символи підстановки у діалоговому вікні **Пошук і заміна** можна використовувати для пошуку полів з типом «Дата/Час», якщо формат, застосований до цих полів, відображає частину дати або всю її у вигляді тексту. Наприклад, можна виконати пошук за допомогою рядка - 10-*брь*-2007 і результати будуть містити назви місяців, у яких є текст «брь» - вересень, жовтень і т. д. Слід пам'ятати, що, оскільки пошук повинен здійснюватися з використанням формату, застосованого до даних, в діалоговому вікні необхідно вибрати параметр - прапорець **З урахуванням формату полів**.

Література:

1. Руденко В.Д. Базы даних в інформаційних системах./За заг.ред. В.Ю.Бикова / навч посібник для студентів педагогічних університетів. – К.:Фенікс, 2010. – 240 с.
2. Бойко В.В., Савинков В.М. Проектирование баз данных информационных систем. – М.:Финансы и статистика.1999. – 371 с.
3. Вейскас Д. Эффективная работа с Microsoft Access: Пер. с англ. – СПб:Питер Ком, 1999. 974 с.

4. Дейт К. Дж. Введение в системы баз данных: Пер. с англ. – 6-е изд. – К.: Диалектика, 1998. – 784 с.
5. Державний стандарт України. Системи оброблення інформації. Бази даних. Терміни і визначення. ДСТУ 2874-94. – К.: Держстандарт України, 1994. – 31 с.
6. Дженигс Р. Использование Microsoft Access: Пер. с англ. – 2-е изд. – К.;М.;СПб.: Изд. Дом «Вильямс», 1998. – 944 с.
7. Коннолли Т., Бегг К., Страчан А. Базы данных: проектирование, реализация и сопровождение: Теория и практика// Пер. с англ.: Учебн. Пособие. – 2-е изд. – М.: Изд. дом «Вильямс», 2000. – 1120 с.
8. Ситник Н.В. Проектування баз і сховищ даних: Навч. Посібник. – К.: КНЕУ, 2004. – 348 с.
9. Хансен Г., Хансен Д. Базы данных: разработка и управление: Пер. С англ.. – М.: БИНОМ, 1999. – 704 с.
10. Хомоненко А. Д., Цыганков В. М., Мальцев М. Г. Базы данных: Учебник для высш. Учеб. Заведений / Под ред.. проф. А. Д. Хомоненко. - СПб. 6 КОРОНА-принт, 2000. – 416 с.