

І.О. ЗАВАДСЬКИЙ

# Основи баз даних

Київ  
ПП І.О. Завадський  
2011

УДК 004.65(075.3)

ББК 32.973-018.2я721

З-13

Рецензенти: П.П. Кулябко, заступник декана факультету кібернетики КНУ ім. Тараса Шевченка, кандидат фіз.-мат. наук, доцент

Д.І. Кожем'яка, учитель інформатики Фінансово-правового ліцею Фінансово-правового коледжу КНУ ім. Тараса Шевченка

*Схвалено Міністерством освіти і науки,  
молоді та спорту України  
(Лист №1.4/18-Г-666 від 20.07.2011)*

Завадський І.О.

З-13 Основи баз даних: [Навч. посіб.] / І.О. Завадський. — К.: Видавець І.О. Завадський, 2011. — 192 с.: іл.  
ISBN 978-966-97182-0-4

Посібник призначено для учнів 10–11 класів та вчителів інформатики загальноосвітніх навчальних закладів. Видання повністю відповідає схваленій МОН України програмі курсу за вибором «Основи баз даних». У ньому розглянуто основи теорії та практичне застосування реляційних баз даних. Базовий програмний продукт — Microsoft Access. Матеріал посібника організовано так, що він може використовуватися з будь-якою популярною версією цієї СКБД: 2003, 2007 або 2010.

**ББК 32.973-018.2я721**

ISBN 978-966-97182-0-4

© І.О. Завадський, 2011

## Передмова

Кваліфіковані користувачі пакету Microsoft Office часто вважають системи керування базами даних черговою офісною програмою, а самі бази даних ставлять у ряду інших документів, таких як презентації чи електронні таблиці. Слід зазначити, що таке ставлення сильно припинює справжню роль інформаційної технології, якій присвячено даний посібник. Бази даних (БД) посідають особливе місце поміж інших галузей інформатики з багатьох причин.

По-перше, сфера їх застосування є найширшою. Практично кожна більш-менш серйозна програма зберігає свої дані в базі. БД використовують представники ледь не кожної професії (часто, однак, не підозрюючи, що вони працюють саме з базами даних). По-друге, бази даних у багатьох інформаційних системах виявляються ключовою ланкою, без котрої система просто «розсиплеться». По-третє, на відміну від усіх інших інформаційних технологій, реляційні бази даних мають під собою міцний і несуперечливий теоретичний фундамент. Організація баз даних — це наука у класичному розумінні цього слова.

Нарешті, вивчення баз даних важливе з причин методичного характеру, адже дослідження та розробка баз даних — це найпростіший спосіб зазирнути на «внутрішню кухню» програмного забезпечення. Навіть ті учні, яким зовсім не дається програмування, цей курс опанують порівняно легко, а розроблювані ними проекти баз даних цілком можна вважати повноцінними прикладними програмами.

Цей посібник, звичайно, дає змогу лише розпочати знайомство з теорією і практикою баз даних. Він відповідає схваленій МОНМС програмі курсу за вибором «Основи баз даних», яка передбачає два варіанти навчання: базовий (15 годин) і розширений (30 годин). Розділи та підрозділи посібника також поділено на ті, що вивчаються в обох варіантах, і ті, які слід вивчати лише в розширеному курсі; останні позначено символом «♦». Символом «\*» позначено складні навчальні вправи і завдання.

Посібник розраховано відразу на три версії СКБД MS Access: 2003, 2007 та 2010. Описи дій, які в цих версіях СКБД виконуються по-різному, виділено спеціальними рамками.

Щодо змістових особливостей посібника насамперед слід зауважити, що значна увага приділяється побудові моделей «сутність-зв'язок», або так званому семантичному моделюванню. Вкрай важливо, щоб ця тема вивчалася до того, як учні почнуть працювати з СКБД, адже робота з нею замулює сутність моделювання предметних областей різноманітними технологічними аспектами.

Крім того, чи не вперше у шкільній навчальній літературі пропонується вивчення основ мови SQL (у розділах розширеної версії). Зазначимо, що без засвоєння цієї теми адекватне уявлення про системи баз даних сформувавши неможливо, адже в основу всіх реляційних СКБД покладено насамперед структуровану мову запитів, а майстри і конструктори — це вторинні візуальні засоби. Зауважимо також, що мова SQL декларативна, а тому опанувати її значно легше, ніж будь-яку мову програмування.

Бажаємо передумовою успішного засвоєння курсу є вміння працювати з табличним процесором, а точніше з такими його засобами, як фільтри, підсумки і функції для роботи з базою даних.

Згідно з програмою курсу за вибором його вивчення має завершуватися виконанням індивідуального або колективного навчального проекту, що полягає у розробці бази даних для заданої предметної області. Описи 7 предметних областей для таких проектів наведено в додатку.

До кожного розділу курсу розроблено електронний урок, що містить, поряд із теоретичним матеріалом, кілька десятків міні-завдань. Їх виконання дозволяє досягти глибокого розуміння сутності технологій, що вивчаються в курсі, і значно полегшує подальшу роботу учнів з СКБД.

У курсі, особливо в розширеній версії, викладено багато малознайомих широкому загалу вчителів інформатики фактів, понять і методів. Тому автор проводить серію онлайн-майстер-класів для учителів.

Бажаємо вам приємної творчої роботи і швидкого опанування однієї з найбільш захоплюючих інформаційних технологій!

**Сайт електронної підтримки курсу:** <http://zavadsky.at.ua>

## Розділ 1

# Основи баз даних

### Повторення

- ◆ Як інформаційна система взаємодіє з інформаційними джерелами та споживачами інформації?
- ◆ Що таке клієнт-серверна мережа?
- ◆ Якими є основні функції табличного процесора?
- ◆ Які завдання з обробки наборів однотипних об'єктів дає змогу виконати табличний процесор?

Ви вже вмiєте працювати з такими прикладними програмами, як системи обробки текстiв, графiчні редактори, табличнi процесори тощо. Кожну з цих програм призначено для вирiшення певного кола завдань, якi важко (хоча iнколи й можливо) реалiзувати за допомогою iнших засобiв. Одне iз завдань, що iх зручно виконувати в середовищi табличного процесора, — це оперування наборами однотипних об'єктiв, поданими у виглядi таблиць, наприклад списком учнiв класу або розкладом руху транспорту. Ви можете сортувати такi набори, фiльтрувати iх, обчислювати пiдсумковi характеристики, застосовувати функцiї для автоматизованого вибирання даних тощо. Проте в табличному процесорi данi зручно обробляти лише в тих таблицях, якi непов'язанi одна з одною. Але так буває далеко не завжди: якщо, скажiмо, в однiй таблицi зберiгається журнал успiшностi, а в iншiй — журнал з результатами медичного обстеження учнiв того ж класу, то цi таблицi є зв'язаними i виконати таке завдання, як обчислення середнього зросту вiдмiнникiв, у табличному процесорi буде доволi проблематично.

Для зберігання і обробки кількох взаємопов'язаних наборів однотипних об'єктів використовують інший інструмент — *системи керування базами даних*, які є основним предметом дослідження в нашому курсі.

## Поняття бази даних

Термін «база даних» почали використовувати в 60-х роках ХХ століття. Існують десятки різних означень цього поняття. Ми будемо дотримуватися такого означення.

**База даних (БД)** — це структурована сукупність даних, які відображують стан об'єктів певної предметної області та зв'язки між ними.

Пояснимо ключові поняття, вжиті в цьому означенні.

*Предметною областю (ПО)* називають сферу застосування конкретної бази даних. Це може бути, наприклад, медицина, освіта або залізничний транспорт. База даних містить відомості лише про ту частину ПО, яка важлива для користувачів бази даних. Так, якщо предметною областю є автомобільні дороги країни і база даних для цієї ПО розробляється для потреб пасажирів, то в ній міститимуться дані про те, яким транспортом можна проїхати з одного населеного пункту в інший, скільки на це знадобиться часу й коштів. Якщо ж база даних орієнтована на водіїв, то в ній має зберігатися інформація про заправні станції, відстані між населеними пунктами, розташування станцій технічного обслуговування, про небезпечні ділянки доріг тощо.

*Об'єктами предметної області* можуть бути підприємства, школи, учні, учителі, книжки та ін. Нагадаємо, що об'єкт має певний набір параметрів, а кожен параметр має значення. Так, параметрами об'єкта «людина» можуть бути прізвище, ім'я, по батькові, рік народження, а їх значеннями — Войцеховський, Микола, Олексійович, 1956. Між деякими об'єктами існують *зв'язки*. Наприклад вислів «людина має собаку» відображає

зміст зв'язку між об'єктами «людина» і «собака», а «машина їде по дорозі» — між об'єктами «машина» і «дорога».

Отже, базу даних можна розглядати як електронний аналог картотеки, де на кожній картці записані відомості про певний об'єкт, а також містяться посилання на інші картки, що представляють зв'язані об'єкти.

*Схему*, або структуру, бази даних слід відрізнити від її *наповнення*. Схема визначає, які **параметри** повинні мати об'єкти, що зберігаються в базі, а наповнення — це **значення** параметрів конкретних об'єктів, які записані в БД на поточний момент. Наприклад, схема БД може бути описана так: зберігаються відомості про ім'я, прізвище, по батькові та рік народження вчителів, а також про назви предметів, які вони викладають. Наповнення цієї БД може бути таким: учитель Пилипчук Олександр Павлович 1964 р. н. читає математику та інформатику, учитель Шестопалов Євген Анатолійович 1937 р. н. читає хімію та біологію.

## Системи керування базами даних

Бази даних — це не різновид програмного забезпечення, а лише документи, з якими оперують спеціальні прикладні програми, що їх називають *системами керування базами даних* (так само, як з електронними таблицями оперують табличні процесори, а з зображеннями — графічні редактори).

**Система керування базами даних (СКБД)** — це програма, що забезпечує можливість створення БД та виконання різноманітних операцій з даними, які в ній зберігаються.

## Переваги використання СКБД

З СКБД зазвичай взаємодіють не лише люди, а й інші прикладні програми. Взагалі, СКБД відіграє роль своєрідної оболонки навколо баз даних, яка встановлює правила роботи з ними будь-яких зовнішніх користувачів, або *клієнтів* (рис. 1.1). Ці правила однакові як для людей, так і для програм.

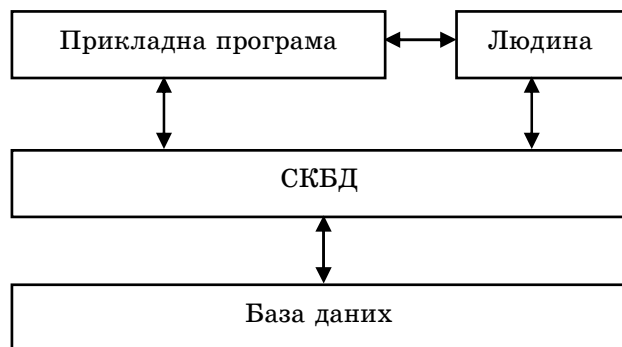


Рис. 1.1. Схема роботи СКБД

Такий підхід означає, що дані в БД більш захищені, ніж у документах інших типів, до яких прикладні програми можуть звертатися безпосередньо. Ви можете малювати що завгодно на растровому зображенні, вводити будь-який текст у текстовий документ або довільні дані в електронну таблицю, але з базою даних таке «не пройде»: введення даних, що не відповідають схемі бази або деяким іншим умовам, буде заблоковано СКБД. Ця властивість даних, що зберігаються в базах, називається *цілісністю*.

Під **цілісністю даних** у базі розуміють їх відповідність схемі БД, а також іншим правилам і умовам, що гарантують несуперечливість даних та їх узгодженість з предметною областю.

Підтримка цілісності в БД важлива з кількох причин, і насамперед тому, що доступ до баз мають програми. Скажімо, якщо в тексті буде зазначено, що певна людина живе на вулиці Київській, а на наступній сторінці — що на вул. Львівській (очевидне порушення цілісності!), то ви якось дасте собі раду з цим фактом, можливо, зрозумієте з контексту, яка інформація є достовірною, а от програма — «істота» нерозумна, і така помилка в даних може призвести до невіправного збою в її роботі. Не слід забувати і про підвищену важливість деяких даних, що

зберігаються в базах, наприклад, даних про суму коштів на картковому рахунку. Очевидно, що некоректність таких даних неприпустима у жодному разі.

Цілісність забезпечується завдяки грамотному проектуванню схеми БД (за це відповідає розробник бази даних), а також завдяки тому, що СКБД забезпечує дотримання *обмежень цілісності* — спеціальних умов, яким мають відповідати коректні дані. Приклади обмежень цілісності: «людина має одне прізвище», «сума на кредитному рахунку не повинна перевищувати 100 000 грн». Створення обмежень цілісності — справа розробника БД.

Важливою перевагою використання СКБД є забезпечення *незалежності даних від програм*. Якщо прикладна програма реалізує алгоритм розв'язання певної задачі, а дані зберігаються в базі, то зміни в структурі даних не впливатимуть на прикладну програму, тобто не призведуть до необхідності змінити її логіку. З іншого боку, зміна логіки прикладної програми не призведе до зміни структури даних — СКБД знов-таки відіграє роль своєрідного «амортизатора».

Варто також зазначити, що СКБД забезпечують *централізоване зберігання інформації*, підвищуючи в такий спосіб її точність та достовірність. Так, у разі використання БД не може виникнути неузгодженості між інформацією про службове становище працівника, що виводиться програмою, призначеною для потреб відділу кадрів, і програмою, яка використовується у бухгалтерії, оскільки відповідні дані зберігаються в одному місці — спільній базі даних. Забезпечення *спільного доступу* до бази кількох клієнтів — ще одна перевага СКБД.

Отже, підтримка цілісності даних, незалежність даних від програм, централізоване зберігання інформації та забезпечення спільного доступу до даних кількох клієнтів — основні переваги використання СКБД порівняно з прямим доступом до даних прикладних програм. Можна назвати й низку інших важливих переваг, проте їх розгляд виходить за межі завдань базового курсу інформатики.

Для допитливих. Бази даних і СКБД — не найдосконаліші засоби зберігання даних. Наприклад, інформацію, що зберігається в пам'яті людини, записати адекватно в БД практично неможливо, оскільки більшість відомостей, які ми пам'ятаємо, є нечіткими, неповними, суперечливими і т. п. Крім того, СКБД не дають змоги робити зі збережених у базі фактів логічні висновки, генерувати нові відомості, приймати рішення тощо. Для вирішення цього комплексу завдань використовуються бази знань та експертні системи. Про них ви можете дізнатися зокрема зі статей «База знань» та «Експертні системи» в українській Вікіпедії.

## Функції СКБД

З самого означення СКБД випливає, що можна виділити дві основні категорії функцій СКБД: функції *визначення даних* — ті, що дозволяють визначити структуру даних у базі, та функції *маніпулювання даними*, які стосуються операцій із самими даними (рис. 1.2). Крім того, СКБД дає змогу тривалий час зберігати дані у цілісному вигляді, захищаючи їх від зловмисних та ненавмисних спотворень, тобто виконувати функцію *збереження даних*, а також ряд інших, менш важливих, функцій, які на рис. 1.2 не вказані.

Зауважте, що на позначення операцій з даними вживають достатньо нестандартні терміни: стосовно СКБД ми кажемо не «введення даних», а «додавання», не «редагування», а «оновлення», не «виведення», а «пошук і вибирання». Це не випадково, адже, наприклад, редагувати дані вміють лише люди, а коли до БД звертається програма, вона може тільки видалити елемент даних цілком та записати на його місце новий, тобто оновити дані, а не відредагувати. Так само, тільки люди здатні вводити дані, а програми їх додають. Нарешті, ми говоримо про пошук та вибирання даних у СКБД, а не про їх виведення, оскільки СКБД зовсім не обов'язково виводить дані у тому вигляді, в якому вони були введені в базу. Скажімо, у базу було

введено дані про учнів та учителів, а клієнт хоче отримати прізвища вчителів, які вчать учня Іванова. Щоб виконати цей запит клієнта, СКБД має спочатку *знайти* вчителів, які викладають у класі цього учня, а потім *вибрати* їхні прізвища.

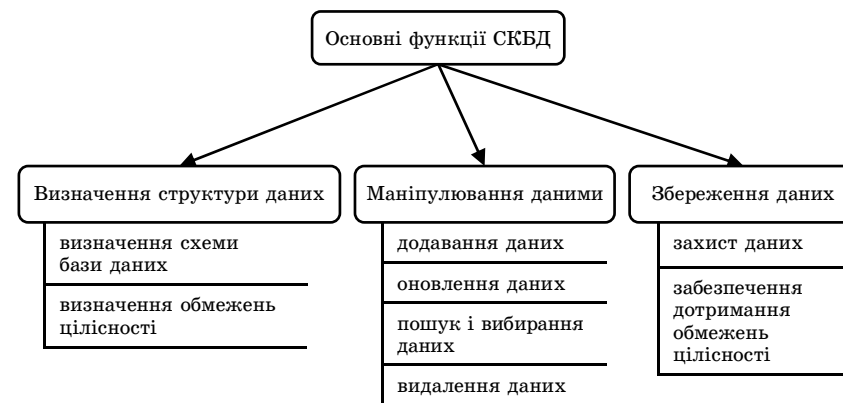


Рис. 1.2. Основні функції СКБД

Зазначимо також, що розглянуту вище функцію забезпечення цілісності слід відрізнити від захисту даних, оскільки в останньому випадку йдеться про захист даних від несанкціонованого доступу. Це означає, що СКБД може надавати доступ до окремих елементів даних лише певним користувачам.

## Ролі користувачів, що працюють з БД

Оскільки схема БД відрізняється від її наповнення, варто розрізнити тих, хто розробляє цю схему, і тих, хто оперує даними в базі. Перших називають *розробниками БД*, других — *користувачами*. Найчастіше користувач взаємодіє не з СКБД безпосередньо, а з прикладною програмою (див. рис. 1.1). Якщо ж людина вводить дані в базу прямо в середовищі СКБД, її називають *оператором бази даних*. Інколи вирізняють ще одну роль — *адміністратора даних*. В обов'язки останнього не входить розробка схеми БД, проте він визначає деякі обмеження цілісності, встановлює права доступу до тих чи інших даних

певних користувачів тощо. Опрацьовуючи матеріал розділів 17–24, ви виконуватимете ролі як розробника СКБД, так і оператора бази даних.

## Моделі даних

Як уже зазначалося, структуру об'єктів та зв'язків між ними, відомості про які зберігаються в БД, визначає схема бази даних. А от правила побудови самої схеми визначаються *моделлю даних*. Крім того, модель даних регламентує, які операції ми можемо виконувати над даними.

**Модель даних** — це система правил, згідно з якими створюють структури даних, здійснюють доступ до даних та змінюють їх.

У більшості сучасних СКБД підтримується *реляційна модель даних*, попередниками якої були *ієрархічна* та *мережева* моделі. Реляційна модель даних, в основу якої покладено математичне поняття *відношення* (англійською — *relation*), вперше була описана у 1970 році англійським кібернетиком Е. Ф. Коддом. У формі відношень у реляційній моделі подаються набори однотипних об'єктів. Відношення зручно зображувати у вигляді таблиці, тому фактично реляційна база даних — це сукупність таблиць, кожна з яких містить набір однотипних об'єктів.

**Для допитливих.** Дуже часто термін «реляційний» помилково інтерпретують як «оснований на зв'язках» і основною особливістю реляційної моделі вважають наявність зв'язків між елементами даних. Насправді «зв'язок» англійською мовою буде «*relationship*», а термін «*relation*», точним перекладом якого є «відношення», в теорії баз даних майже еквівалентний терміну «таблиця». Отже, «реляційний» фактично означає «табличний».

Протягом останніх 15–20 років стали досліджуватися постреляційні моделі, найбільш перспективною з яких вважається

*об'єктно-орієнтована модель даних*. Утім вона значною мірою відтворює ідеологію ієрархічної моделі, її розвиток відбувається повільно, тож на ринку СКБД, скоріш за все, ще тривалий час домінуватимуть реляційні системи.

Уважно переглянувши означення моделі даних, ми побачимо, що цим терміном охоплюються два поняття: правила визначення структури даних та правила маніпулювання даними. В усіх відомих сьогодні моделях правила маніпулювання даними реалізовано у вигляді спеціальних мов, на яких клієнти мають описувати *запити* до СКБД. Йдеться про так звані *мови маніпулювання даними*, які ще неформально називають мовами запитів. Часто до складу моделі даних включають ще й іншу мову, яка дозволяє описувати схеми баз даних і називається *мовою визначення даних*.

У реляційній моделі найбільш популярною мовою є SQL (від англ. Structured Query Language — мова структурованих запитів), яка поєднує в собі засоби мови визначення даних і мови маніпулювання даними. Можливість описувати запити до баз даних цією мовою надають усі відомі на сьогодні реляційні СКБД. Мова SQL не є мовою програмування, вона *декларативна*, тобто дозволяє користувачу описати, **що** він хоче отримати, не описуючи, **як** саме комп'ютер має обчислити потрібний результат. Тому писати запити мовою SQL значно легше, ніж програми будь-якою мовою програмування, але і це не завжди потрібно робити, адже деякі СКБД надають візуальні засоби конструювання запитів і серед них — СКБД MS Access, з якою ви працюватимете, опановуючи наступні розділи.

**Для допитливих.** Мова SQL, хоча і є найпопулярнішою, не зовсім точно відповідає постулатам реляційної моделі даних, за що була піддана різкій критиці самим автором моделі Е. Коддом. Ще до появи SQL він запропонував дві інші мови маніпулювання даними — *реляційну алгебру* та *реляційне числення*, засоби яких включено до складу SQL, але з деякими неточностями.

## Висновки

- ◆ База даних (БД) — це структурована сукупність даних, які відображують стан об'єктів певної предметної області та зв'язки між ними. Предметною областю (ПО) називають сферу застосування конкретної бази даних.
- ◆ Схема визначає, які параметри повинні мати об'єкти, що зберігаються в базі, а наповнення — це значення параметрів конкретних об'єктів, відомості про які записані в БД на поточний момент.
- ◆ Система керування базами даних (СКБД) — це програма, що забезпечує можливість створення БД та виконання різноманітних операцій з даними, які в ній зберігаються.
- ◆ Під цілісністю даних у базі розуміють їх відповідність схемі БД, а також іншим правилам і умовам, що гарантують несуперечливість даних та їх узгодженість з предметною областю.
- ◆ Обмеження цілісності — це умови, яким мають відповідати коректні дані.
- ◆ Основні переваги використання СКБД — підтримка цілісності даних, забезпечення незалежності даних від програм, централізоване зберігання інформації та надання спільного доступу до даних кільком клієнтам.
- ◆ Основні функції СКБД полягають у наданні клієнтам засобів для визначення структури даних, маніпулювання даними (їх додавання, оновлення, пошуку, вибирання й видалення), та зберіганні даних у цілісному вигляді.
- ◆ Дві категорії людей, що працюють з БД, — це розробники, які створюють схему БД, та користувачі БД, які виконують операції з даними в базі.
- ◆ Моделлю даних називають систему правил, за якими створюють структури даних, здійснюють доступ до даних та змінюють їх.
- ◆ Реляційна база даних — це сукупність таблиць, кожна з яких містить набір однотипних об'єктів.
- ◆ Маніпулювання даними здійснюють за допомогою спеціальних мов. У реляційній моделі найпопулярнішою є мова SQL, яка надає засоби як для маніпулювання даними, так і для їх визначення.

## Питання для роздумів

1. Чи можуть існувати бази даних з однаковою схемою, але різним наповненням? А з різними схемами, але однаковим наповненням?
- 2\*. Чи є перевагою використання СКБД прискорення доступу прикладних програм до даних? Відповідь аргументуйте.

## Завдання для досліджень

1. Припустимо, що є відомості про багатоповерхові будинки, а саме про адреси, за якими вони розташовані, кількість та номери квартир на кожному поверсі, кількість кімнат у квартирах, а також про мешканців цих квартир (прізвище, ім'я, по батькові, дата народження). Виділіть у цій предметній області кілька наборів однотипних об'єктів та вкажіть параметри об'єктів у кожному наборі.
2. Які ще функції має СКБД, крім тих, що зазначені на рис. 1.2? Знайдіть потрібну інформацію в Інтернеті.
3. Схема БД має такий вигляд, як показано нижче. Для якої предметної області може бути побудована така база даних та як можуть називатися таблиці в ній? Між якими об'єктами в цій базі можуть існувати зв'язки та яким є зміст цих зв'язків? Заповніть кожну таблицю даними про два-три об'єкти.

Вид	Прізвище	Дата народження	Стать	Вага

№	Площа	Тип	Дата	Час відкриття	Час закриття	Кількість відвідувачів



## Розділ 2

# Модель «сутність-зв'язок»

### Повторення

- ◆ Стан чого відображує база даних?
- ◆ З чого складається предметна область?
- ◆ Чим відрізняється схема бази даних від її наповнення?
- ◆ Що таке цілісність даних?

Із попереднього розділу ви знаєте, що база даних відображає стан об'єктів певної предметної області та зв'язки між ними. Розробник бази даних, проектуючи її схему, встановлює, які параметри мають ці об'єкти. Крім того, він має визначити, які зв'язки можуть існувати між об'єктами. Інакше кажучи, розробник будує *модель* предметної області, яку в теорії баз даних називають *моделлю «сутність-зв'язок»*. Це найперше завдання, яке виконують ще до того, як схема бази даних буде створена в СКБД, і саме над ним ми працюватимемо сьогодні.

### Об'єкти, сутності, зв'язки

Як уже наголошувалося, будь-яка предметна область містить об'єкти. Наприклад, у предметній області «школа» об'єктами можуть бути учителі Михайлюк Дмитро Семенович та Петрова Ніна Володимирівна, класи 10А і 11Б тощо. При цьому класи 10А і 11Б є *однотипними* об'єктами (оскільки вони обидва є класами), а клас 10А і учитель Михайлюк Дмитро Семенович — об'єктами різного типу. Замість фрази «однотипні об'єкти» кажуть також, що об'єкти належать до однакової *сутності*, наприклад учителі Михайлюк Дмитро Семенович та

Петрова Ніна Володимирівна належать до сутності «учитель», а класи 10А та 11Б — до сутності «клас». Критерій того, чи належать об'єкти до однієї сутності, дуже простий: такі об'єкти повинні мати однаковий набір параметрів.

**Сутність у предметній області — це множина об'єктів, які мають однаковий набір параметрів.**

Об'єкти, що належать до однієї сутності, мають не лише однаковий набір параметрів, а й однотипні зв'язки з об'єктами інших сутностей. Наприклад, будь-який клас може бути зв'язаний з учнем зв'язком «учень вчиться у класі», а з учителем — зв'язком «учитель викладає у класі». Тому можна казати, що зв'язки властиві не лише об'єктам, а і сутностям. Твердження «учень вчиться у класі» та «учитель навчає клас» — приклади зв'язків саме між сутностями. (Прикладом зв'язку між об'єктами є твердження «учень Шпак Максим вчиться у 10А класі».) Так само можна казати, що не лише об'єкт, а й сама сутність має певний набір параметрів. Наприклад, сутність «учень» має такі параметри, як прізвище, ім'я, стать, зріст, дату народження тощо.

Тепер ми готові точно визначити призначення моделі «сутність-зв'язок».

**Модель «сутність-зв'язок» призначено для графічного зображення сутностей певної предметної області, їх параметрів та зв'язків між ними.**

**Для допитливих.** Модель «сутність-зв'язок» називають також *ER-моделлю* (від англ. Entity-Relationship), а процес її побудови — *ER-моделюванням*, або *семантичним моделюванням*. «Семантичний» означає «змістовий», і це слово підкреслює, що під час ER-моделювання ми виявляємо зміст сутностей у предметній області та зв'язків між ними.

## Завдання 2.1

Назвіть кілька сутностей з предметних областей «залізниця», «адміністративний устрій», «водопостачання». Які параметри мають ці сутності? Які між ними можуть існувати зв'язки?

## Графічні позначення в моделі «сутність-зв'язок»

Сутності в моделі «сутність-зв'язок» позначаються прямокутниками, всередині яких записуються їхні назви. Назви параметрів сутностей (у теорії реляційних баз даних їх називають *атрибутами*) записуються під прямокутниками сутностей і підкреслюються. Від прямокутника сутності проводять вертикальну лінію вздовж усіх атрибутів. Як приклад на рис. 2.1 зображено сутність *Учитель* із атрибутами *паспорт*, *прізвище*, *ім'я*, *по батькові*, *стать* та *спеціальність*.

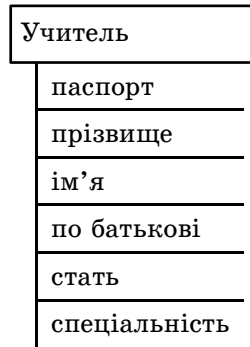


Рис. 2.1. Зображення сутності *Учитель* у моделі «сутність-зв'язок»

Зв'язки між сутностями позначаються ромбами, що з'єднуються з прямокутниками. Всередині ромба записують дієслово або словосполучення, що визначає зміст зв'язку. Якщо записано лише дієслово, над ним часто малюють стрілку, що показує, в якому порядку дієслово сполучає іменники-сутності. Так, на рис. 2.2 зображено зв'язок між сутностями *Учитель* та *Клас*. Стрілка означає, що саме *Учитель* навчає *Клас*.

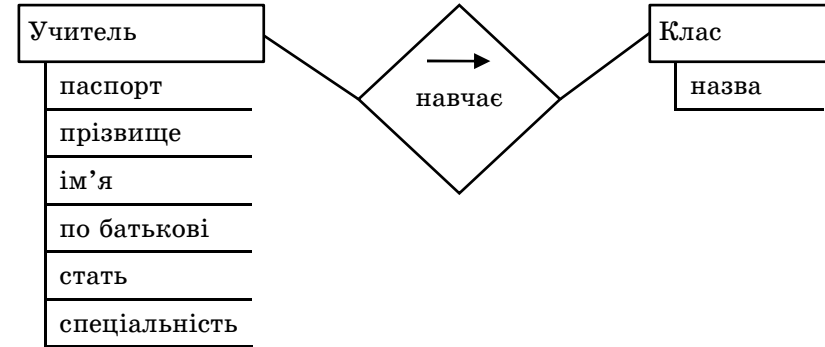


Рис. 2.2. Зв'язок між сутностями *Учитель* та *Клас*

ставимо зворотне запитання: у скількох класах може навчатися кожен учень? Відповідь також очевидна: в одному. Все це означає, що клас зв'язано з учнем зв'язком «*один-до-багато*».

На моделі «сутність-зв'язок» позначимо цей зв'язок словом «*вчиться*»: учень вчиться у класі. Біля лінії, що з'єднує ромб зв'язку з прямокутником «*Клас*», ми маємо записати символ «1», адже учень вчиться в *одному* класі. А біля лінії, що з'єднує ромб зв'язку з прямокутником «*Учень*», ми маємо записати символ «∞» (нескінченність, багато), адже у класі вчиться *багато* учнів (рис. 2.3).

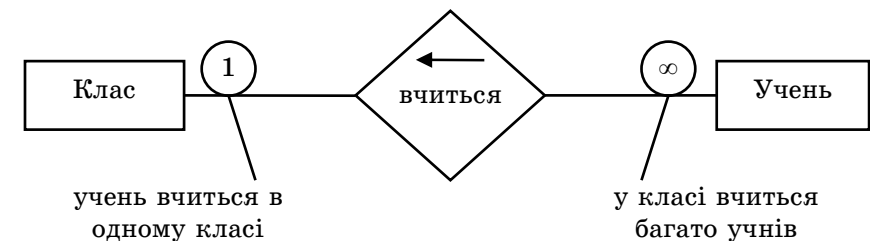


Рис. 2.3. Зображення зв'язку «один-до-багато»

Загалом зв'язки між сутностями бувають трьох типів:

- ◆ «один-до-багато»;

- ◆ «один-до-одного»;
- ◆ «багато-до-багатьох».

Щоб зв'язок між сутностями А і В віднести до одного з цих трьох типів, слід відповісти на два запитання:

- ◆ Зі скількома об'єктами сутності А може бути зв'язаний об'єкт сутності В?
- ◆ Зі скількома об'єктами сутності В може бути зв'язаний об'єкт сутності А?

Як відповіді на ці запитання визначають типи зв'язків і як ці зв'язки позначаються графічно, стане зрозуміло після уважного розгляду табл. 2.1.

Таблиця 2.1. Різновиди зв'язків

Тип зв'язку	Графічне позначення	Зі скількома об'єктами А може бути зв'язано об'єкт В?	Зі скількома об'єктами В може бути зв'язано об'єкт А?
«один-до-багатьох»		з одним	з багатьма
«один-до-одного»		з одним	з одним
«багато-до-багатьох»		з багатьма	з багатьма

Віднесення зв'язку до одного з трьох розглянутих типів називають класифікацією зв'язку за множинністю, а самі позначення «один» та «багато» — множинностями зв'язку з того чи іншого боку. Є й інші способи класифікації зв'язків, деякі з котрих ми розглянемо в наступному розділі. А зараз наведемо приклади зв'язків різної множинності.

**Зв'язок «учитель навчає клас».** У кожному класі можуть викладати багато вчителів. З іншого боку, кожен учитель може викладати в багатьох класах. Таким чином, зв'язок «учитель навчає клас» має тип «багато-до-багатьох» (рис. 2.4).

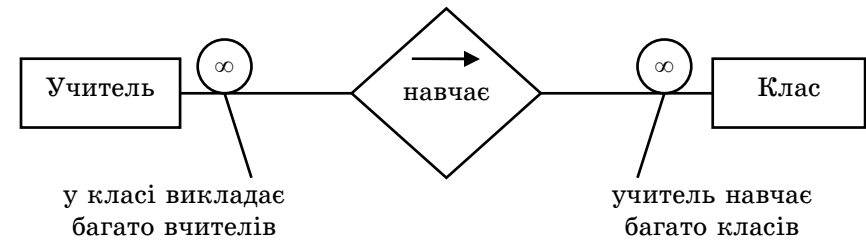


Рис. 2.4. Зображення зв'язку «багато-до-багатьох»

**Зв'язок «учитель є класним керівником».** Учитель може керувати лише одним класом, а у класі може бути тільки один класний керівник. Таким чином, зв'язок «учитель є класним керівником» має тип «один-до-одного» (рис. 2.5).

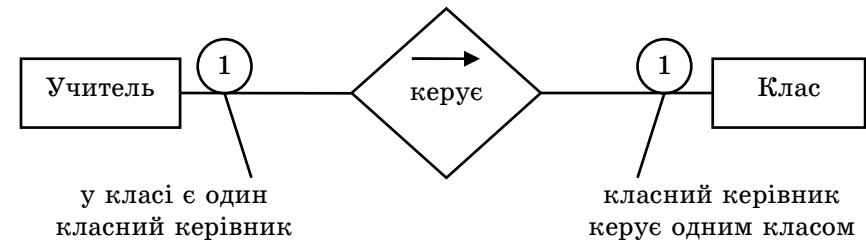


Рис. 2.5. Зображення зв'язку «один-до-одного»

**Для допитливих.** Крім трьох розглянутих типів множинності бувають й інші. Наприклад, батьки зв'язані з дітьми зв'язком типу «два-до-багатьох»: кожна дитина має двох батьків, а в кожного батька може бути скільки завгодно дітей.

## Завдання 2.2

Визначте типи зв'язків та зобразіть зв'язки графічно:

- ◆ курка несе яйця;
- ◆ у яйці є жовток;
- ◆ річка протікає територією держави;
- ◆ у команді 11 футболістів.

## Ключові атрибути

Якщо ви відкриваєте рахунок у банку, купляєте квиток на літак або вступаєте у вищий навчальний заклад, вас неодмінно попросять вказати номер і серію свого паспорта. Чому саме так, чому недостатньо, скажімо, назвати своє прізвище, ім'я та по батькові? Річ у тім, що номер та серія паспорта *однозначно ідентифікують* особу, оскільки ніколи не повторюються. Про ім'я, прізвище та по батькові цього не можна сказати, і в принципі можлива ситуація, коли Іванчук Дмитро Васильович прийде до банку і забажає зняти кошти з рахунку іншого Іванчука Дмитра Васильовича. Якби особа, якій належить рахунок, визначалася за прізвищем, ім'ям та по батькові, в описаний спосіб можна було б вкрасти гроші.

Атрибут, значення якого не може повторюватися, називають *ключовим* або просто *ключем*. Ключі відіграють у базах даних украй важливу роль, адже саме за їх допомогою СКБД ідентифікує об'єкти. Ви вже зрозуміли, що для сутності *Учитель* таким атрибутом може бути номер та серія паспорту (назвемо його просто *паспорт*). Об'єкти сутності *Клас* у предметній області «школа» можна ідентифікувати за атрибутом *назва*, адже в одній школі не може бути двох класів з однаковою назвою.

Зазначимо, що ключові атрибути на моделі «сутність-зв'язок» позначаються символом «\*» (рис. 2.6).

Децю складніша ситуація з ключем сутності *Учень*. Більшість учнів не мають паспорта, а прізвище та ім'я в різних учнів можуть бути однаковими. Можна припустити, що в одній шко-

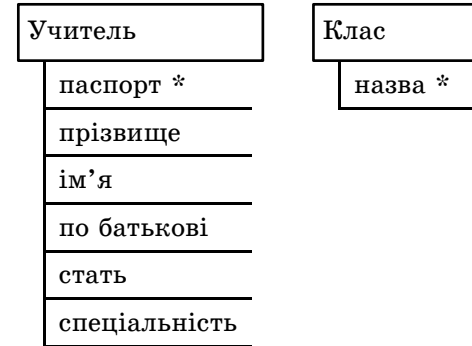


Рис. 2.6. Ключі сутностей *Учитель* та *Клас*

ключових атрибутів — ні.

Проте в дуже рідкісних випадках припущення про неіснування двох учнів з однаковим прізвищем та ім'ям, що народилися в один день, може не справджуватися і, крім того, оперувати в СКБД з ключем, який містить три атрибути, незручно. Тому часто вихід знаходять у додаванні штучного ключового атрибута, який можна назвати *код* (рис. 2.7, б). У такому випадку учні школи в базі даних ідентифікуватимуться за спеціальним, штучно створеним, кодом.

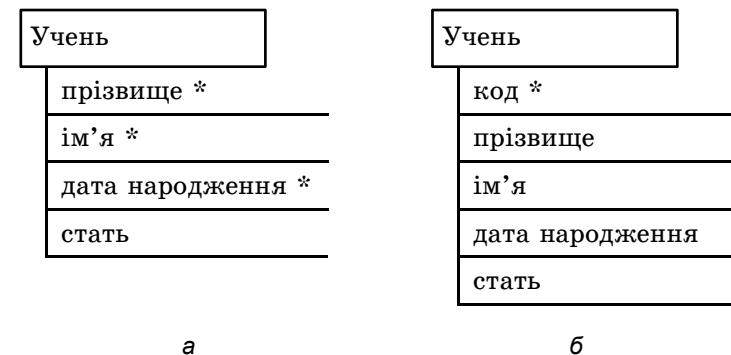


Рис. 2.7. Ключі сутності *Учень*: а — складений ключ, що містить три атрибути; б — штучно доданий ключовий атрибут

### Завдання 2.3

Після назв сутностей у дужках вказано їхні атрибути. Визначте, які ключі мають ці сутності. Якщо з наведених атрибутів утворити ключ неможливо, вкажіть, які атрибути слід додати.

- ◆ Холодильник (модель, об'єм, рік випуску, серійний номер).
- ◆ Ріка (назва, довжина).
- ◆ Місто (назва, кількість населення, рік заснування, область, держава).
- ◆ Авіарейс (аеропорт відльоту, аеропорт посадки, дата відльоту).

### Побудова моделі «сутність-зв'язок»

У попередніх підрозділах розділу 17 ми фактично побудували модель «сутність-зв'язок» для предметної області «школа». Опишемо детально ту частину предметної області, якої стосується ця модель.

*Про учнів певної школи відомі їхні імена, прізвища, дати народження та стать. Кожен учень вчиться у певному класі. У класах викладають вчителі. Потрібно зберігати відомості про номер та серію паспорта, прізвище, ім'я та по батькові, а також про спеціальність вчителів. Деякі вчителі є класними керівниками.*

Модель «сутність-зв'язок» описаної частини предметної області подано на рис. 2.8.

Зауважте, що в розглянутій частині предметної області «школа», на додаток до трьох зображених зв'язків між сутностями, можна виявити й інші, зокрема «учитель навчає учня». Проте зображувати цей зв'язок на моделі «сутність-зв'язок» неможна, оскільки він є наслідком вже показаних зв'язків: які вчителі навчають того чи іншого учня, зрозуміло з того, у якому класі цей учень вчиться та які вчителі в цьому класі викладають.

Щоб краще зрозуміти, чому зв'язок «учитель навчає учня» зайвий, припустимо, що ми його створили. Припустимо також, що учитель Петрова Ніна Володимирівна викладає у класах

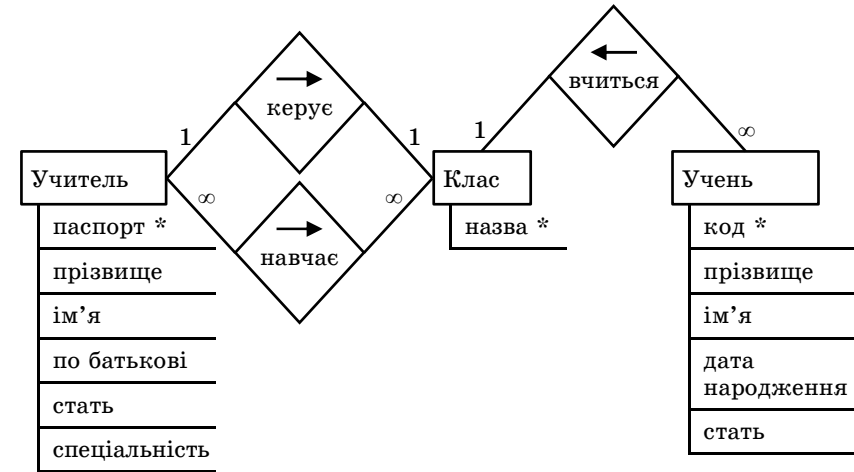


Рис. 2.8. Модель «сутність-зв'язок» для предметної області «школа»

10А і 11А, тобто зв'язана з ними зв'язком «учитель викладає в класі». Нам нічого не завадить, однак, зв'язати цього вчителя і з учнями 11Б класу через новостворений зв'язок «учитель навчає учня». Вийде, що вчитель навчає учнів того класу, у якому не викладає, що суперечить здоровому глузду. Якщо ж модель «сутність-зв'язок» має такий вигляд, як на рис. 2.8, то описана ситуація неможлива в принципі.

Зазначимо, що додатковий зв'язок між учителем і учнем стане доцільним, якщо ми забажаємо відображувати відомості про індивідуальні позакласні заняття. Однак у наведеному вище описі предметної області про це нічого не сказано.

### Головний принцип семантичного моделювання

Міркування, подібні до наведених у попередньому підрозділі, дозволяють сформулювати головний принцип, якому має відповідати правильно побудована модель «сутність-зв'язок» предметної області, або, інакше кажучи, *головний принцип семантичного моделювання*. Але перш ніж його формулювати, зро-

бимо одне важливе зауваження. А саме, зазначимо, що в будь-якій базі даних зберігаються не просто відомості, а *факти*. Наприклад, той факт, що вчитель із паспортом СН 410268 має прізвище Михайлюк, ім'я Дмитро, по батькові Семенович, стать чоловічу та спеціальність математик, може зберігатися в таблиці *Учителі*, а той факт, що Михайлюк Дмитро Семенович викладає в 11А класі, зберігатиметься завдяки зв'язку «учитель викладає у класі».

З описаної точки зору модель «сутність-зв'язок» встановлює правила збереження фактів у базі даних і головний критерій якості цієї моделі є таким.

**Головний принцип семантичного моделювання.** Модель «сутність-зв'язок» повинна дозволити зберігання будь-якого факту лише в одному місці.

Зауважимо, що введення до зображеної на рис. 2.8 моделі зв'язку «учитель навчає учня» порушить цей принцип: той факт, що певний учитель навчає певного учня, зберігатиметься і в цьому зв'язку, і у зв'язках «учитель викладає в класі» та «учень навчається в класі», разом узятих.

**Для допитливих.** Головний принцип семантичного моделювання дуже вдало формулюється через поняття надлишковості інформації (згадайте матеріал 9-го класу): модель «сутність-зв'язок» потрібно проектувати так, щоб інформація, яка зберігатиметься у спроектованих за цією моделлю БД, мала якомога меншу надлишковість.

## Висновки

- ◆ Сутність у предметній області — це множина об'єктів, які мають однаковий набір параметрів.
- ◆ Модель «сутність-зв'язок» призначено для графічного зображення сутностей певної предметної області, їхніх параметрів та зв'язків між ними.

- ◆ Сутності в моделі «сутність-зв'язок» позначаються прямокутниками, всередині яких записуються їхні назви. Назви параметрів сутностей записують під прямокутниками сутностей і підкреслюють. У теорії реляційних баз даних такі параметри називають атрибутами.
- ◆ Зв'язки між сутностями позначають ромбами, які з'єднують з прямокутниками. Всередині ромбу записують дієслово або словосполучення, яке визначає зміст зв'язку. Коли записано лише дієслово, над ним малюють стрілку, що показує, у якому порядку дієслово сполучає іменники-сутності.
- ◆ Зв'язки між сутностями бувають трьох типів: «один-до-багатьох»; «один-до-одного»; «багато-до-багатьох». Віднесення зв'язку до одного з цих типів називають його класифікацією за множинністю.
- ◆ Сутність А зв'язана з сутністю В зв'язком «один-до-багатьох», якщо кожен об'єкт А може бути зв'язаний з довільною кількістю об'єктів В, а кожен об'єкт В — лише з одним об'єктом А.
- ◆ Сутність А зв'язана з сутністю В зв'язком «один-до-одного», якщо кожен об'єкт А може бути зв'язаний тільки з одним об'єктом В, а кожен об'єкт В — лише з одним об'єктом А.
- ◆ Сутність А зв'язана з сутністю В зв'язком «багато-до-багатьох», якщо кожен об'єкт А може бути зв'язаний з довільною кількістю об'єктів В, а кожен об'єкт В — з довільною кількістю об'єктів А.
- ◆ Ключем називають атрибут або набір атрибутів, значення яких дозволяють однозначно ідентифікувати об'єкти певної сутності.
- ◆ Головний принцип семантичного моделювання полягає в тому, що модель «сутність-зв'язок» повинна дозволити зберігання будь-якого факту лише в одному місці.

## Завдання для самостійного виконання

Побудуйте моделі «сутність-зв'язок» для описаних далі предметних областей. Визначте:

- ◆ сутності;
- ◆ атрибути сутностей;
- ◆ ключі;
- ◆ зв'язки між сутностями;
- ◆ типи зв'язків.

1. **Предметна область «автомобілі».** Про кожен автомобіль відомо: його реєстраційний номер в ДАІ, рік випуску, марка, а також хто є власником автомобіля. Про власника відомо: прізвище та ім'я, а також номер прав водія. Щодо кожної марки автомобіля відома її назва, об'єм та потужність двигуна, а також тип автомобіля (седан, хетчбек, джип тощо). На автомобілі виписують страхові поліси, у яких зазначається термін початку та завершення дії, тип страховки, страхова сума та назва компанії-страхувальника. На один автомобіль може бути виписано багато страхових полісів.

2. **Предметна область «вулиці та будинки».** Щодо кожної вулиці потрібно зберігати відомості про її назву та довжину, а також про те, які інші вулиці вона перетинає. Про кожен будинок відомо: його номер, кількість поверхів та квартир, а також на якій вулиці він розташований.

**ВКАЗІВКА.** У моделі «сутність-зв'язок» цієї предметної області одна з сутностей буде зв'язана сама з собою.

3\*. **Предметна область «зовнішнє незалежне оцінювання».** Потрібно зберігати відомості про склад тестів зовнішнього незалежного оцінювання. Кожен тест стосується певного предмета, проводиться в певному році та складається з низки запитань. Кожне запитання має формулювання, а також кілька варіантів відповідей. Один із варіантів є правильним.

**ВКАЗІВКА.** У моделі «сутність-зв'язок» цієї предметної області між двома тими самими сутностями потрібно буде створити два різних зв'язки.

4\*. **Предметна область «залізнична мережа».** Залізнична мережа розглядається як сукупність залізничних ділянок, кожна з яких має певну довжину і сполучає дві вузлові станції або вузлову та кінцеву станції. Ділянку посередині

не можуть перетинати інші ділянки. Про кожну станцію відома її назва, а також на якій ділянці вона розташована. Вузлові станції можуть належати кільком ділянкам.

**ВКАЗІВКА.** У моделі «сутність-зв'язок» цієї предметної області варто використати зв'язок типу «два-до-багатьох», а також два різних зв'язки між двома тими самими сутностями.

5\*. **Предметна область «розклад занять».** Про класи певної школи відомі їхні назви та яка кількість учнів навчається в кожному класі, про навчальні дисципліни — назви та тип (гуманітарна, природнича тощо), а про вчителів — прізвища та номери паспортів. Необхідно зберігати відомості про те, який учитель в якому класі яку дисципліну викладає.

**ВКАЗІВКА.** Для адекватного моделювання цієї предметної області потрібно створити зв'язок відразу між трьома сутностями. Спробуйте самостійно здогадатися, які бувають типи зв'язків між трьома сутностями та за яким критерієм ці типи визначати.

6\*. **Предметна область «векторні зображення в MS Word».** Векторне зображення складається з геометричних фігур: ліній, кіл, прямокутників, трикутників, ромбів. Кожна з фігур має координати точки прив'язки на площині малюнка. Крім того, замкнені фігури мають заливку, кола — радіус, прямокутники — координати протилежного до точки прив'язки кута тощо. Фігури можуть об'єднуватися в групи. Група — це також фігура, яка має координати точки прив'язки на площині малюнка. Саме векторне зображення характеризується місцем у документі та способом обтікання текстом.

**ВКАЗІВКА.** Якщо кожну з сутностей Лінія, Коло, Прямокутник і т. д. зв'язувати із сутностями Зображення та Група, модель вийде надто захарашеною зв'язками, а в сутностях фігур повторюватимуться ті самі атрибути. Натомість краще створити сутність Фігура. Подумайте, якою буде множинність її зв'язків з сутностями Лінія, Коло, Прямокутник тощо.

## Питання для роздумів

1. Вкажіть, які ключові атрибути можуть мати сутності На-селений пункт, Країна, Автобус, Дачна ділянка, Книга.
2. Що з переліченого нижче може бути сутністю у предметній області «школа»: класна кімната, номер класної кімнати, директор, навчальна дисципліна, школа?
3. Одним із атрибутів сутності Клас є кількість учнів. Чи варто відображати цей атрибут у моделі «сутність-зв'язок», поданій на рис. 2.8? Відповідь аргументуйте.
- 4\*. Які зв'язки в моделях, побудованих у завданнях для самостійного виконання, є обов'язковими?
- 5\*. Чи може сутність мати більше одного ключа? Якщо відповідь «так», наведіть приклад, якщо «ні» — аргументуйте. Зауважте, що йдеться не про той випадок, коли до складу одного ключа входить кілька атрибутів, а саме про той, коли є кілька ключів.
- 6\*. Чому у завданні для самостійного виконання № 5 недостатньо створити два або три зв'язки, кожен із яких зв'язуватиме лише дві сутності?

## Розділ 3°

# Поглиблене семантичне моделювання

### Повторення

- ◆ Наведіть приклад зв'язку між об'єктами та зв'язку між сутностями.
- ◆ Які бувають типи зв'язків за множинністю та як ці типи визначати?
- ◆ Сформулюйте головний принцип семантичного моделювання.
- ◆ Який атрибут або набір атрибутів називають ключем сутності?

У попередньому розділі ми розглянули 7–8 фундаментальних понять семантичного моделювання, таких як сутність, атрибут, зв'язок між двома сутностями, множинність зв'язку та ін. Загалом під час моделювання предметних областей застосовують більше двох сотень концепцій, лише деякі з котрих можна зобразити графічно. Сьогодні ми навчимося використовувати ще кілька ідей та понять, які за важливістю та частотою застосування можна вважати наступними за фундаментальними. Як приклад продовжимо розглядати предметну область «школа», поступово додаючи до неї нові сутності та розширюючи її на всі школи певного регіону.

### Обов'язковість зв'язків

Згадаємо попередній розділ: щоб визначити множинність зв'язку, потрібно було відповісти на запитання: «Зі скількома об'єктами сутності А може бути зв'язаний об'єкт сутності В?». Сло-



ва «може бути» у ньому вжито не випадково, оскільки в загальному випадку наявність зв'язку між сутностями означає, що об'єкт лише **може**, а не повинен бути зв'язаний з іншим об'єктом або об'єктами. Проте інколи об'єкти саме **повинні** брати участь у зв'язках, і такі зв'язки називають **обов'язковими** з того чи іншого боку. Наприклад, кожен учень повинен вчитися в певному класі й тому зв'язок «учень вчиться у класі» є обов'язковим з боку сутності *Учень*. Обов'язковість зв'язку позначається подвійною лінією (рис. 3.1).

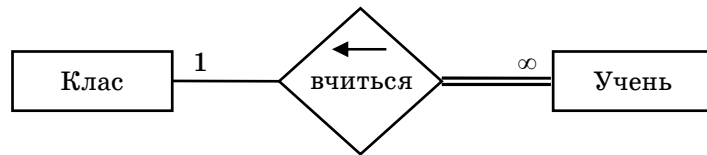


Рис. 3.1. Зображення обов'язковості зв'язку в моделі «сутність-зв'язок»

Зв'язок «учитель є класним керівником» обов'язковий для класу, адже немає класів без класних керівників. Можна помітити й інші «обов'язковості»: у класі обов'язково мають бути учні, учитель має обов'язково викладати принаймні в одному класі та в кожному класі обов'язково має викладати принаймні один вчитель. Однак, як стане зрозуміло з наступних розділів, підтримувати три останні вимоги обов'язковості складно з технічної точки зору. Легко реалізувати лише обов'язковість зв'язку «один-до-багатьох» з боку «багато» та зв'язку «один-до-одного» з якогось одного боку. З урахуванням цього обов'язковість зв'язків у моделі предметної області «школа» слід подати так, як показано на рис. 3.2.

## Слабкі сутності

Розширимо предметну область з однієї школи на всі школи певного міста чи району. Тоді у предметній області з'явиться сутність *Школа*. Її атрибутами можуть бути назва та адреса, причому назва є ключем, оскільки в одному населеному пункті чи районі назви шкіл не можуть повторюватися. Зауважте, що коли ми розглядали одну школу, вона була не сутністю пред-

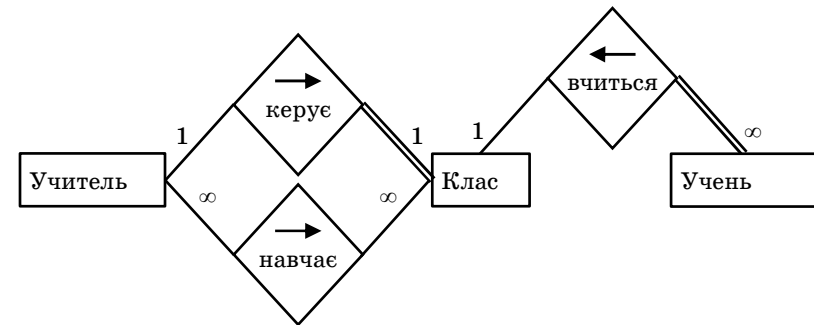


Рис. 3.2. Обов'язкові зв'язки у моделі «сутність-зв'язок» для предметної області «школа»

метної області, а самою предметною областю і тому на діаграмі «сутність-зв'язок» не відображалася.

Очевидно, що між школою та класом існує зв'язок «школа містить клас». Школа може містити багато класів, але кожен клас належить тільки одній школі, тому множинність цього зв'язку — «один-до-багатьох» (рис. 3.3). Звичайно, сутність *Школа* бере участь і в інших зв'язках, наприклад «учень вчиться у школі» та «вчитель працює у школі». Але відображення зазначених зв'язків на моделі порушило б головний принцип семантичного моделювання.

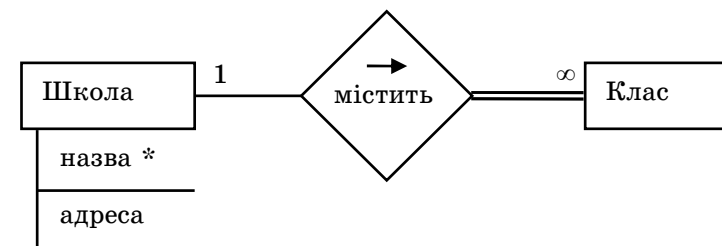


Рис. 3.3. Відображення сутності *Школа* на моделі «сутність-зв'язок»

## Завдання 3.1

Поясніть, чому відображення на моделі «сутність-зв'язок» зв'язків «учень вчиться у школі» та «вчитель працює у школі» по-

рушує головний принцип семантичного моделювання. Якщо вам важко виконати завдання, прочитайте уважно текст після означення головного принципу в попередньому розділі.

У результаті розширення предметної області на кілька шкіл виникає проблема з ключем сутності Клас. Адже тепер об'єктами цієї сутності будуть класи різних шкіл, а в різних школах назви класів можуть повторюватися. Тобто сам по собі атрибут назва вже не є ключем сутності Клас. Однак неважно помітити, що клас однозначно ідентифікує пара атрибутів назва класу + назва школи, адже в межах однієї школи назви класів не повторюються. Хоча атрибут назва школи не належить сутності Клас, ця сутність може його «запозичити» з метою доповнення ключа.

Сутність, якій не вистачає власних атрибутів для утворення ключа, називається *слабкою* і позначається на моделі «сутність-зв'язок» подвійним прямокутником. Її ключ утворюють з деяких її власних атрибутів, а також ключів інших сутностей. Зв'язки слабкої сутності з тими сутностями, що доповнюють її ключ, називаються *підтримувальними* та позначаються подвійними ромбами (рис. 3.4).

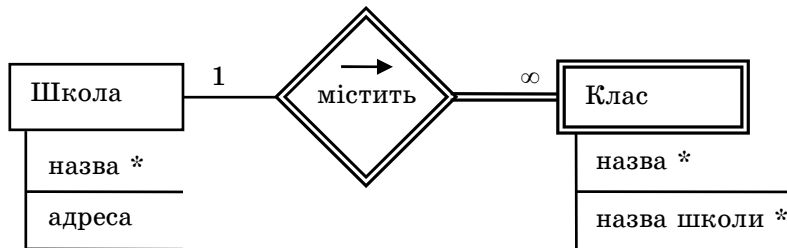


Рис. 3.4. Слабка сутність і підтримувальний зв'язок

### Завдання 3.2

На рис. 3.5 зображено модель предметної області «галузь промисловості», але без зазначення ключів сутностей. З'ясуйте, яка сутність або сутності слабкі, які їхні зв'язки є підтримувальними

та якими атрибутами слід ці сутності доповнити, щоб сформулювати їхні ключі.

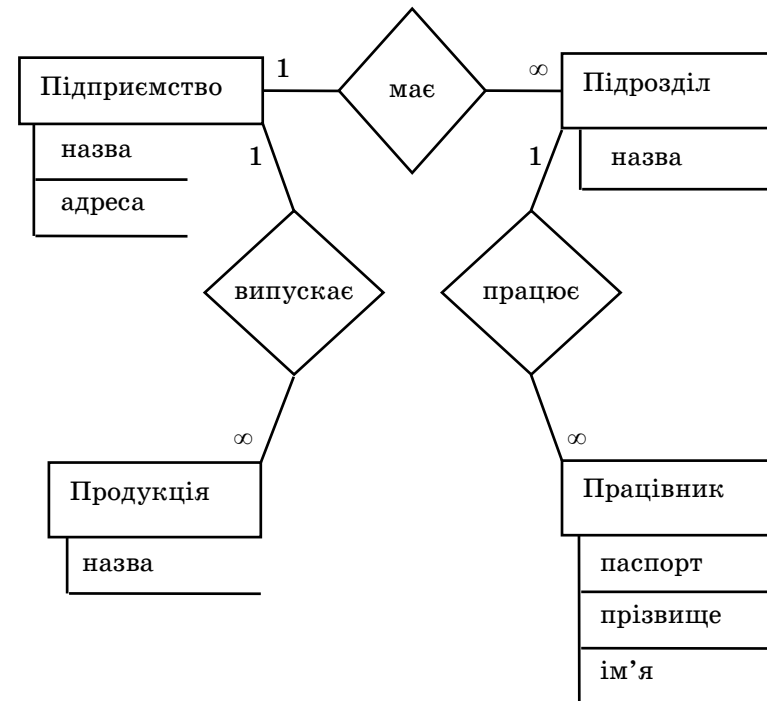


Рис. 3.5. Модель «сутність-зв'язок» предметної області «галузь промисловості»

### Зв'язок «є»

Додамо до нашої предметної області сутність Директор. Вона матиме ті самі атрибути, що й сутність Учитель: номер паспорта, прізвище, ім'я, по батькові, стать та спеціальність. Більше того, директор має всі ті зв'язки, що й учитель: він може викладати в класах, а також може бути класним керівником (хоча це трапляється рідко). Така подібність директора до вчителя пояснюється дуже просто, адже директор є вчителем. Інакше кажучи, сутності Учитель та Директор зв'язані специфічним

зв'язком «є», який називається також зв'язком «загальний тип-різновид». Директора можна розглядати як різновид учителя, а вчителя вважати більш загальним, ніж директор, типом педагогічного працівника.

Зв'язок «є» на ER-моделі позначається не ромбом, а рівнобедреним трикутником, вершина якого вказує на загальний тип (рис. 3.6). Множинність будь-якого зв'язку «є» становить «один-до-одного». Справді, кожен директор є одним учителем, а кожен учитель може бути не більш, ніж одним директором. Цей зв'язок завжди обов'язковий з боку сутності-різновиду і необов'язковий з боку загального типу. Так, у нашому прикладі кожен директор обов'язково повинен бути вчителем, але не кожен учитель повинен бути директором.

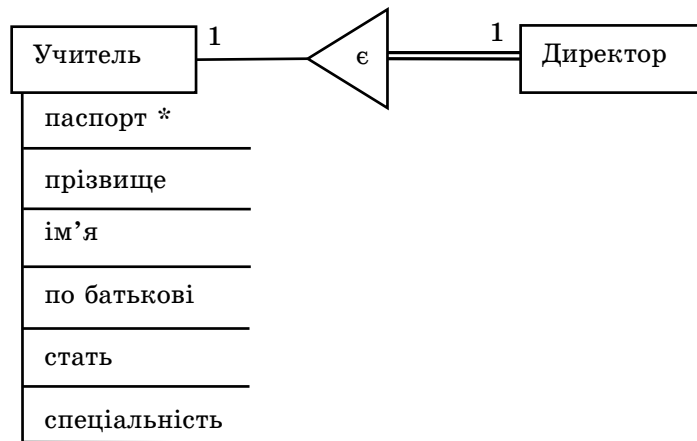


Рис. 3.6. Зв'язок «є»

Задасмося питанням: з якою метою для зв'язка «є» введено спеціальне позначення? Річ у тім, що воно дозволяє не креслити повторно атрибути та зв'язки для сутності-різновиду, якщо вони вже накреслені для загального типу. Подивіться на рис. 3.6: хоча директор має всі ті атрибути і зв'язки, що й учитель, креслити їх немає потреби, адже сутність-різновид **завжди** має всі ті атрибути та зв'язки, що й сутність-загальний тип і сама наявність між директором та вчителем зв'язку «є» вже озна-

чає, що директор має номер паспорта, прізвище, ім'я, по батькові, стать та спеціальність, викладає у класах та може бути класним керівником.

Однак сутність-різновид може мати й інші зв'язки та атрибути, крім тих, які вона успадковує від загального типу. Зокрема директор, на відміну від учителя, зв'язаний зі школою зв'язком «керує» (рис. 3.7). Множинність цього зв'язку — «один-до-одного», оскільки будь-який директор керує однією школою, і кожною школою керує один директор.

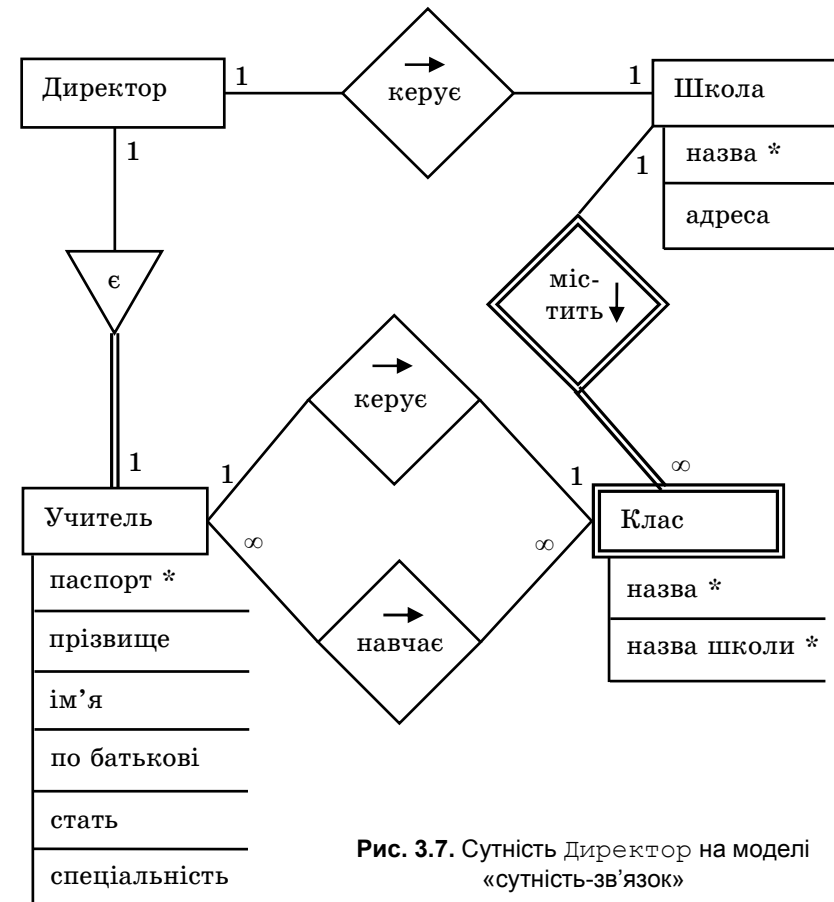


Рис. 3.7. Сутність Директор на моделі «сутність-зв'язок»

Для допитливих. Зв'язки «є» можуть утворювати ієрархічні дерева. Наприклад, на додаток до того, що директор є різновидом учителя, можна зазначити, що директор є різновидом педагогічного працівника. Різновидом пед. працівника є також методист, а різновидом учителя — заступник директора. Педагогічний працівник, у свою чергу, є різновидом працівника взагалі, а працівник взагалі — різновидом громадянина. У такий спосіб утворюється дерево, зображене на рис. 3.8. Його коренем є сутність Громадянин, а листям — сутності Директор, Заступник директора, Методист. Проміжні гілки (не корінь і не листя) — це сутності Працівник і Учитель. На відміну від звичайного дерева, корінь ієрархічного дерева прийнято розташовувати вгорі, а листя — внизу.

Отже, директор запозичує всі атрибути та зв'язки у вчителя, вчитель — у педагогічного працівника, педагогічний працівник — у працівника взагалі, а той — у громадянина. Тобто сутність запозичує атрибути та зв'язки всіх сутностей, розташованих на шляху від неї до кореня дерева ієрархії. Звідси випливає, що кожен атрибут або зв'язок слід розташовувати якомога ближче до кореня — тоді його не доведеться повторювати в сутностях нижчих рівнів. Так, у зображеній на рис. 3.8 ієрархії атрибути паспорт, прізвище та ім'я можна розташувати в корені, оскільки їх мають усі громадяни, атрибут трудова книжка мають усі працівники (але не всі громадяни), а педагогічний стаж — усі педагогічні працівники.

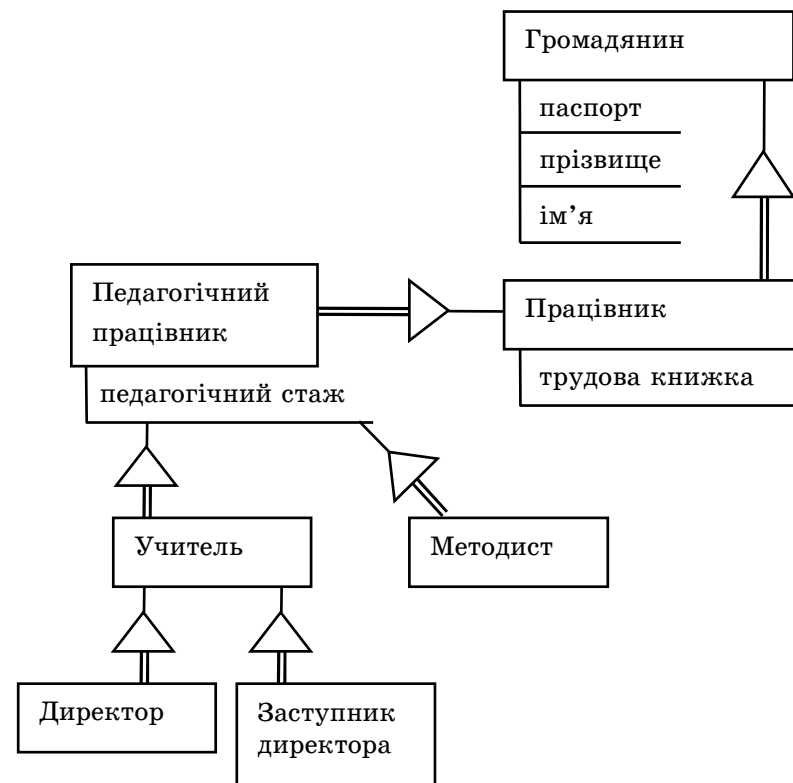


Рис. 3.8. Ієрархічне дерево зв'язків «є»

### Завдання 3.3

Нижче перелічено пари сутностей. Вкажіть, у яких парах існують зв'язки «є» та зобразіть їх графічно.

- ◆ керівник та підлеглий;
- ◆ тварина і ссавець;
- ◆ книга і журнал;
- ◆ носій інформації та флешка;
- ◆ трамвай і пасажир;
- ◆ автомобіль і колесо;
- ◆ автомобіль і транспортний засіб.

### Зв'язок між кількома сутностями

Тепер додамо до предметної області «школи» сутність Предмет. Зрозуміло, що вона буде зв'язана з сутністю Учитель, адже вчителі викладають предмети, та Класи, оскільки класи предмети вивчають. Легко побачити, що множинність обох цих зв'язків буде «багато-до-багатьох», а отже на моделі «сутність-зв'язок» вони відобразатимуться так, як показано на рис. 3.9, а.

Однак така модель є недосконалою. Річ у тім, що вона не дозволяє з'ясувати, який учитель у якому класі який предмет викладає. Справді, припустимо, що Сошко Катерина Миколаївна є

вчителем біології та хімії, але в 10А класі вона викладає тільки біологію і цей факт нам потрібно зберігати в базі даних. Зв'язок «учитель викладає предмет» дає змогу вказати, що ця вчителька викладає біологію та хімію, зв'язок «клас вивчає предмет» — те, що 10А клас вивчає біологію, хімію та інші предмети, а зв'язок «учитель навчає клас» — те, що К.М. Сошко веде заняття в 10А та інших класах. Чи впливає з цих трьох фактів те, що Катерина Миколаївна Сошко викладає біологію 10А класу? Поміркуйте добре над цим запитанням. Відповідь буде негативною, оскільки жоден факт не вказує на те, що вчителька Сошко викладає 10А класу саме біологію, а не хімію.

Таким чином, для адекватного відображення предметної області недостатньо трьох зв'язків, кожен з яких з'єднує дві сутності (рис. 3.9, а). Зв'язок «вчитель викладає предмет у класі» має з'єднувати відразу три сутності і зображується він так, як показано на рис. 3.9, б. Зазначимо, що зв'язки, які з'єднують три сутності, називаються *тернарними*, а ті, які з'єднують лише дві сутності, — *бінарними*.

Тернарний зв'язок характеризується трьома множинностями, і щоб визначити їх, потрібно дати відповіді на три запитання, у нашому прикладі такі.

- ◆ Скільки класів може відповідати кожній парі «учитель + предмет»? Очевидно, що багато, адже той самий учитель може викладати той самий предмет у багатьох класах. Отже, біля дужки, що з'єднує ромб тернарного зв'язку з прямокутником сутності Клас, ставимо символ «∞».
- ◆ Скільки предметів може відповідати кожній парі «клас + учитель», тобто скільки предметів може читати учитель в одному класі? Відповідь — багато, тому що, скажімо, Катерина Миколаївна Сошко у деякому класі може читати і біологію, і хімію. Символ «∞» записуємо біля сутності Предмет.
- ◆ Скільки вчителів може відповідати кожній парі «клас + предмет»? Відповідь також «багато», оскільки деякі предмети, наприклад іноземну мову, в тому самому класі може читати декілька вчителів. Таким чином, символ «∞» слід записати і біля сутності Учитель.

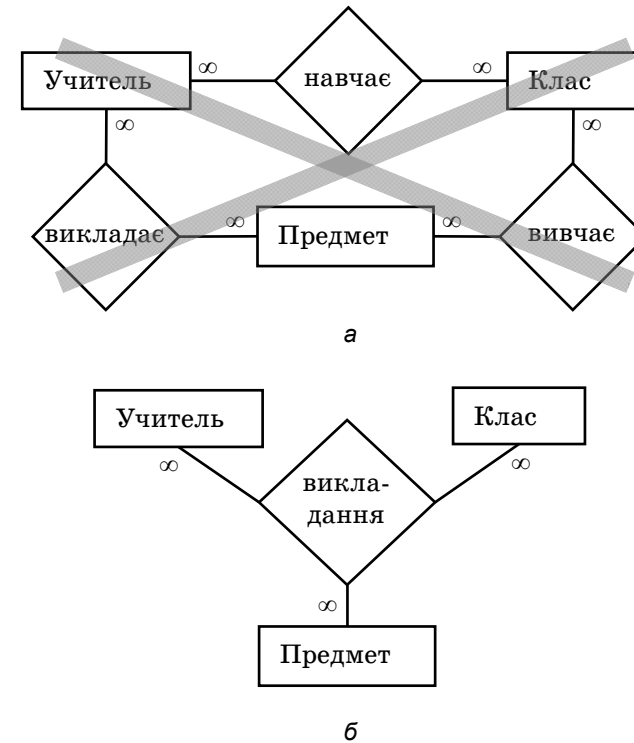


Рис. 3.9. Зображення зв'язку між учителем, предметом і класом: а — недосконала модель; б — правильна модель

Тернарні зв'язки слід створювати з великою обережністю, оскільки дуже часто вистачає двох бінарних зв'язків там, де, здається, існує зв'язок між трьома сутностями. Наприклад, маклери купляють акції компаній на біржі. Може здатися, що вислів «маклер купляє акцію компанії» відповідає тернарному зв'язку (рис. 3.10, а). Однак насправді достатньо лише двох бінарних зв'язків (рис. 3.10, б), оскільки кожна акція належить тільки одній компанії і дізнавшись, яку акцію придбав маклер, ми автоматично дізнаємось і про компанію, акція якої була куплена. Тернарний зв'язок у даному випадку помилковий, адже дозволяє одну ту саму акцію зв'язувати з різними

компаніями в контексті різних покупок. Звичайно, цей зв'язок порушує і головний принцип семантичного моделювання, дозволяючи багаторазово зберігати відомості про те, якій компанії належить акція.

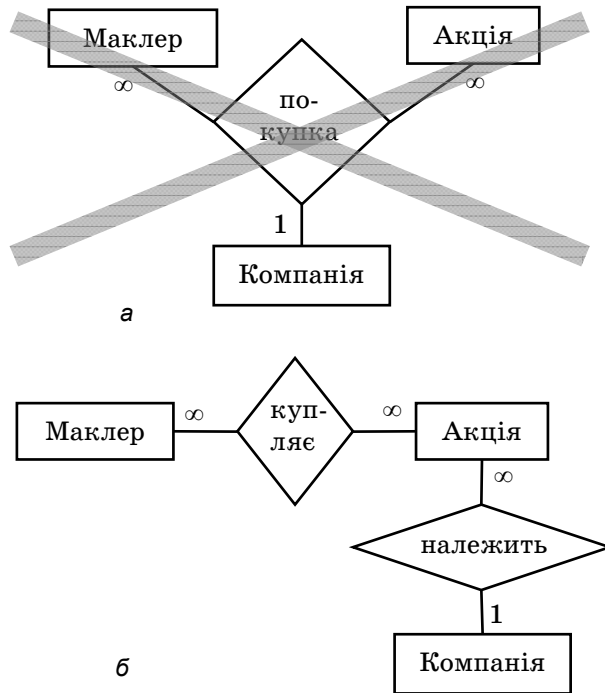


Рис. 3.10. Моделювання зв'язку між маклером, акцією та компанією в предметній області «біржа»: а — помилкове; б — правильне

### Завдання 3.4

Нижче описано два фрагменти предметних областей. Побудуйте для кожного з них модель «сутність-зв'язок», створивши, якщо потрібно, тернарні зв'язки. Визначте множинність усіх зв'язків. Атрибути сутностей можна не вказувати.

- а) **Предметна область «виробництво і торгівля».** Є відомості про різноманітні товари, а також про підприємства, що їх

виробляють (для кожного товару є лише один виробник). Крім того, є дані про покупців, які укладають угоди з виробниками на закупівлю їхніх товарів. Одна угода може стосуватися лише одного товару.

- б) **Предметна область «програмування».** Програмісти беруть участь у проектах з розробки програмного забезпечення. В одному проекті може використовуватися кілька мов програмування, але кожним програмістом — одна. У різних проектах програміст може програмувати різними мовами.

Для допитливих. Інколи трапляються зв'язки між чотирма, п'ятьма або більшою кількістю сутностей. Зв'язок між  $n$  сутностями називається  $n$ -арним зв'язком, або зв'язком степені  $n$ .

### Зв'язок сутності самої з собою

Припустимо, що нам потрібно зберігати відомості про подружні зв'язки між учителями. Для цього слід створити зв'язок «шлюб» між учителем та учителем (рис. 3.11). Один кінець цього зв'язку відповідатиме чоловіку, інший — дружині. Множинність цього зв'язку має бути «один-до-одного», адже в людини може бути лише один чоловік або жінка. Оскільки не кожен учитель повинен одружуватись на вчителі чи виходити заміж за вчителя, зв'язок не є обов'язковим з жодного боку.

Варто також унеможливити моделювання одностатевих шлюбів, тобто заборонити з'єднувати через зв'язок «шлюб» учителів однакової статі. Суто графічних засобів для зображення такої заборони не існує, проте її можна формалізувати в текстовому вигляді. Для цього насамперед потрібно дати найменування *полюсам зв'язку*. Поліус зв'язку — це його одна сторона, або лінія, що з'єднує зв'язок з сутністю. Один поліус зв'язку «шлюб» слід назвати чоловік, інший — дружина. Тоді заборону одностатевих шлюбів можна сформулювати за допомогою двох умов: чоловік.стать = 'ч' та дружина.стать = 'ж'.

Ці вирази є обмеженнями цілісності (згадайте перший розділ), які слід записати біля зв'язку (рис. 3.11). Загалом на рис. 3.11 зображено остаточну модель «сутність-зв'язок» предметної області «школи», у якій відображено всі сутності та зв'язки, розглянуті нами раніше.

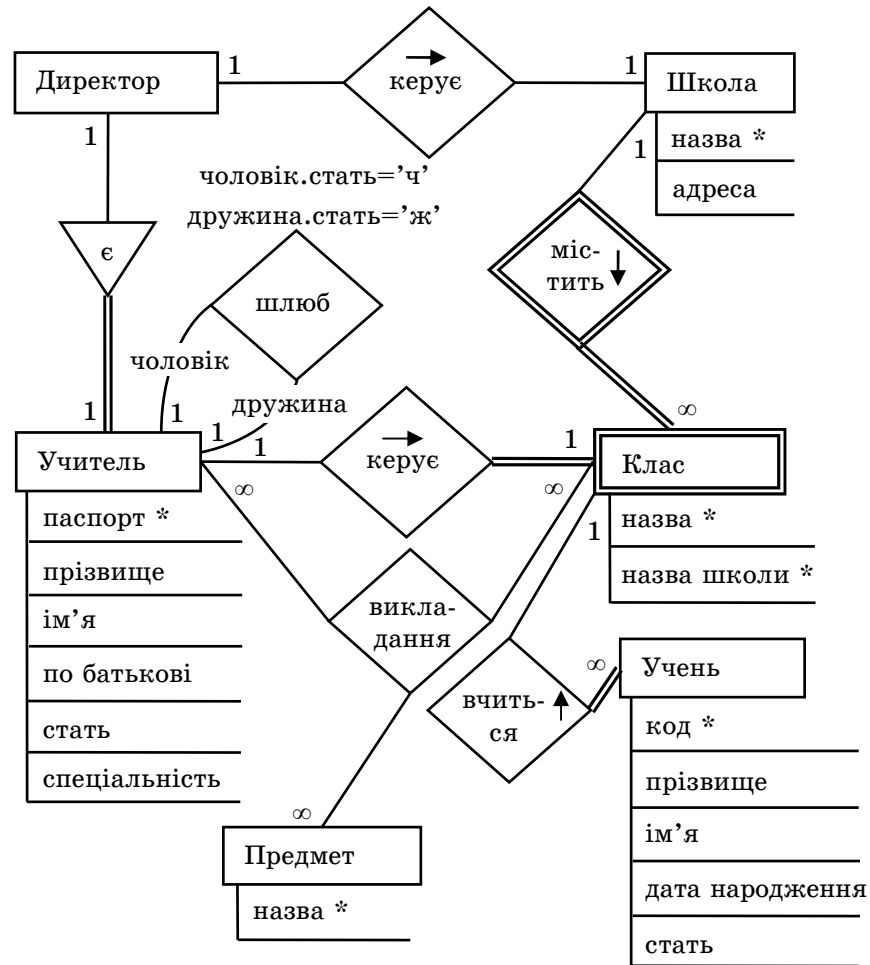


Рис. 3.11. Остаточний вигляд моделі «сутність-зв'язок» предметної області «школи»

### Завдання 3.5

Є сутність Людина з атрибутами прізвище, ім'я, стать, вік. Створіть зв'язок, що відповідав би відношенню між людьми «батьки-діти». Запишіть чи зобразіть обмеження, властиві цьому зв'язку: батьків не може бути більше, ніж двоє, дитина має бути молодшою за батька і матір, стать батьків різна.

### Висновки

- ♦ Зв'язок між сутностями А і В називають обов'язковим з боку сутності А, якщо кожен об'єкт цієї сутності повинен брати участь у зв'язку. На моделі «сутність-зв'язок» обов'язкові зв'язки позначають подвійними лініями.
- ♦ Сутність називається слабкою, якщо їй не вистачає власних атрибутів для утворення ключа. Її ключ утворюють з деяких її власних атрибутів, а також ключів інших сутностей. Зв'язки слабкої сутності з тими сутностями, що доповнюють її ключ, називаються підтримувальними та позначаються подвійними ромбами, а сама слабка сутність — подвійним прямокутником.
- ♦ Сутності А і В з'єднані зв'язком «є», що називається також зв'язком «загальний тип – різновид», якщо сутність В — це різновид сутності А. Зв'язок «є» має множинність «один-до-одного» і є обов'язковим з боку сутності-різновиду. Він позначається рівнобедреним трикутником, вершина якого спрямована до загального типу.
- ♦ Зв'язки між двома сутностями називаються бінарними.
- ♦ Якщо відносини між трьома сутностями неможливо адекватно зобразити за допомогою кількох бінарних зв'язків, усі три сутності з'єднують одним зв'язком, що називається тернарним і має три множинності.
- ♦ Сутність може бути зв'язана сама з собою.

### Завдання для самостійного виконання

Нижче описано фрагмент предметної області «кіноіндустрія». Побудуйте для нього модель «сутність-зв'язок». Визначте сутності, їхні атрибути та ключі, зв'язки між сутностями, тип, множинність і обов'язковість кожного зв'язку.

- ◆ Є кіностудії, що мають назви та адреси. Кіностудією керує президент, прізвище та ім'я якого відомі. Кіностудія може мати підрозділи (наприклад, творче об'єднання документальних фільмів, художніх фільмів, підрозділ технічної підтримки тощо), для кожного з яких відома кількість працівників.
- ◆ Всі кінострічки характеризуються назвою, тривалістю, роком випуску, бюджетом. Крім того, можна виділити жанри стрічок: мультфільми, що, в свою чергу, бувають ляльковими або мальованими; художні фільми, у яких задіяні актори, а також документальні фільми, для яких вказано тематику. Для будь-якої стрічки є одна студія-власник, але, крім неї, брати участь у виробництві картини може ще декілька студій.
- ◆ Про актора відомі його прізвище, ім'я, рік народження та стать. Треба зберігати відомості про те, які ампула має актор у кожному з фільмів, в яких він грає (таких ампула може бути декілька). Щодо кожного ампула відома його назва і текстовий опис особливостей. На кожен фільм актор підписує зі студією-власником фільму окремий контракт.
- ◆ Необхідно зберігати відомості про зіркові шлюби (чоловік та дружина – актори), а також про акторські династії (принаймні один з батьків актора є актором).

## Питання для роздумів

1. Для кожної з наведених трійок сутностей визначте, які між сутностями існують зв'язки. Якщо серед перелічених сутностей є слабкі, вкажіть їх та їхні ключі.
  - ◆ птах, горобець, голуб;
  - ◆ сеанс, кінотеатр, глядач;
  - ◆ магазин, товар, покупець;
  - ◆ принтер, комп'ютер, комп'ютерна мережа.
- 2\*. Припустимо, що потрібно зберігати відомості про те, скільки годин на тиждень викладає кожен учитель той чи інший предмет у певному класі. Де на моделі «сутність-зв'язок», зображеній на рис. 3.9, б, слід накреслити атрибути кількості годин?

## Розділ 4


# Операції з таблицями

## Повторення

- ◆ Що є основним об'єктом у реляційній базі даних?
- ◆ Що таке ключ?
- ◆ Які функції виконує розробник бази даних, а які — її користувач?
- ◆ Для чого призначено систему керування базами даних?

Після того як для предметної області побудовано модель «сутність-зв'язок», можна переходити до роботи з системою керування базами даних. Найперше завдання, що його ви як розробник БД маєте виконати в системі, — це створення бази даних та проектування її схеми у відповідності до моделі «сутність-зв'язок». Кожній сутності цієї моделі має відповідати таблиця в базі даних. Сьогодні ви навчитеся створювати таблиці та налаштовувати їхні параметри. Крім того, почнете працювати з базою даних як користувач, увівши інформацію про декілька об'єктів. Всі ці дії ви виконуватимете у найпопулярнішій серед початківців СКБД Microsoft Access.

## Створення бази даних у СКБД MS Access

Система керування базами даних Microsoft Access входить до складу пакета прикладних програм Microsoft Office. Її ярлик має вигляд ключа  (що символізує, скоріш за все, ключ до сховища, де зберігаються дані), а запускають цю програму так само, як і інші офісні застосунки: за допомогою меню **Пуск** або ярлика на робочому столі.



## MS Access 2003

Документ СКВД Microsoft Access, тобто базу даних, створюють, як і документи в інших офісних програмах, за допомогою кнопки **Создать** або команди **Создать** меню **Файл**. Коли ви клацнете цю кнопку або виконаєте команду, справа у вікні програми відобразиться область завдань **Создание файла**, де буде запропоновано кілька способів створення бази даних (рис. 4.1).

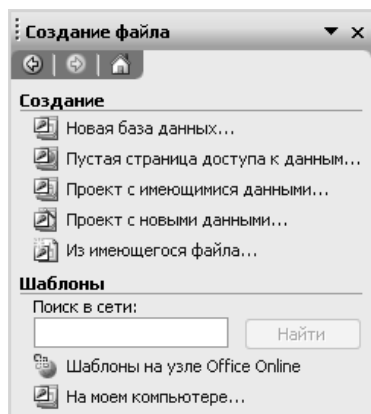


Рис. 4.1. Область завдань **Создание файла**

Нас цікавитиме лише один спосіб: **Новая база данных**, що дозволяє створити порожню базу даних без жодної таблиці.

Після клацання посилання **Новая база данных** буде відображено вікно **Файл новой базы данных**, у якому потрібно вибрати папку, де зберігатиметься файл БД, ввести ім'я файлу та клацнути кнопку **Создать**. Файл бази даних Microsoft Access 2003 має розширення **mdb** та ось таку піктограму:

Коли ви виконаєте описані вище дії, на екран буде виведено головне вікно бази даних (рис. 4.2). Це

означає, що базу даних створено та відкрито.

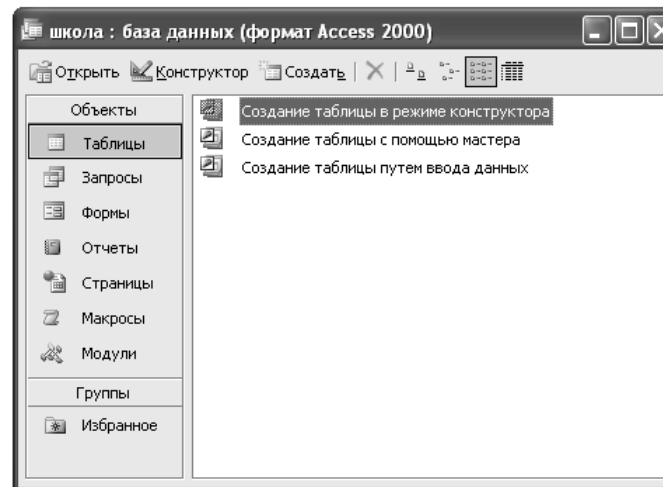


Рис. 4.2. Головне вікно бази даних у MS Access 2003

## MS Access 2007/2010

Для створення бази даних використовують значок **Новая база данных** у вікні **Приступая к работе** або команду **Создать** кнопки **Office**. Після виконання цієї команди буде відображено область завдань **Новая база данных** (рис. 4.3), де потрібно ввести ім'я файлу у відповідне поле та клацнути кнопку **Создать**. Після створення БД буде відразу створено та відкрито таблицю, яка має стандартну назву **Таблица1**. Файл бази даних має розширення **accdb**.

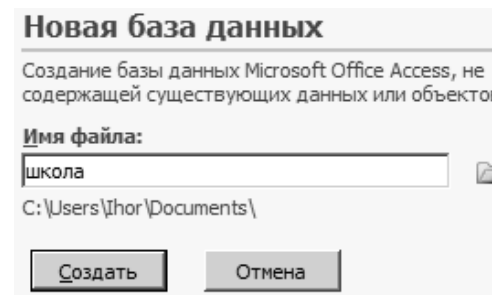


Рис. 4.3. Область завдань **Новая база данных**

### Завдання 4.1

Створіть у вказаній вчителем папці базу даних **школа**.

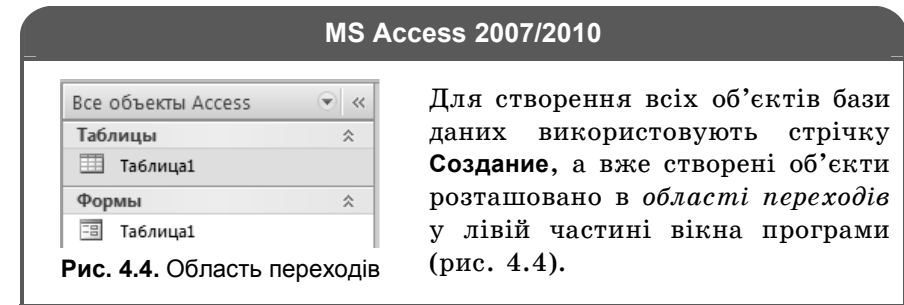
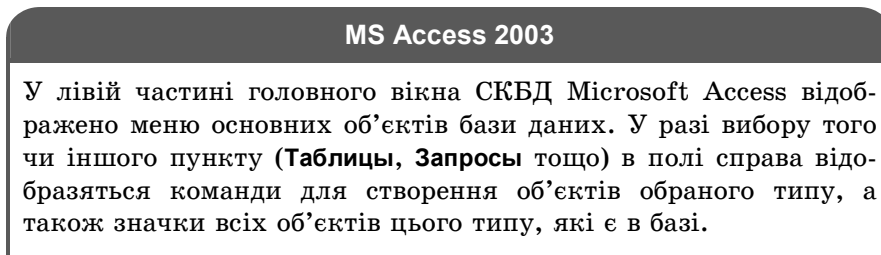
### Основні об'єкти бази даних Microsoft Access

Усього в базах даних Microsoft Access є сім основних різновидів об'єктів, з чотирма з яких ми працюватимемо.

- ◆ **Таблицы (Таблиці)** — головний об'єкт бази даних. Як уже згадувалося в розділі 1, усі дані в реляційній БД зберігаються в таблицях. Усі інші об'єкти БД — допоміжні й призначені для автоматизації різноманітних операцій з даними.
- ◆ **Запросы (Запити)** — компонент, що зустрічається майже в будь-якій реляційній базі даних. Призначені для автоматизації пошуку даних за різноманітними критеріями, а також для додавання, оновлення й видалення даних.
- ◆ **Формы (Форми)** — діалогові вікна, за допомогою яких користувач може вводити дані в таблиці. Підвищують зручність додавання даних.
- ◆ **Отчеты (Звіти)** — макети аркушів паперу, на яких відображаються дані з таблиць і запитів у спосіб, визначений розробником БД. Використання звітів сприяє підвищенню гнучкості відображення даних.

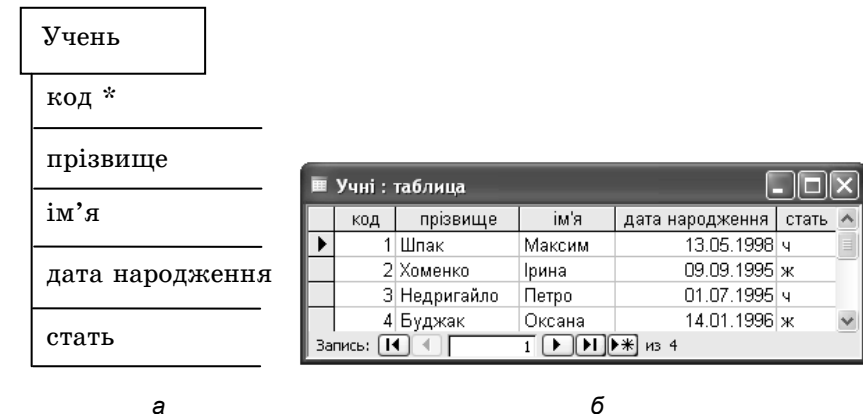
Підкреслимо, що з цих чотирьох компонентів два — таблиці і запити — є елементами самої реляційної моделі даних, а отже, і будь-якої реляційної бази даних, незалежно від того, у якій СКБД її створено. Натомість форми і звіти — це компоненти, специфічні для баз даних Microsoft Access. Переважна більшість реляційних СКБД не надає засобів для створення форм і звітів; це завдання, як правило, покладається на розробників прикладних програм, що використовують бази даних.

Опишемо, де розміщуються і як створюються названі об'єкти в СКБД MS Access.



### Створення таблиць

Кожній сутності в моделі «сутність-зв'язок» має відповідати таблиця в реляційній базі даних. Назва сутності збігається з назвою таблиці з тією лише відмінністю, що назву таблиці записують, як правило, у множині. Атрибутам сутності відповідають стовпці таблиці, які називають також *полями*, а інформацію про кожен об'єкт сутності записують в окремому рядку таблиці, який в реляційних БД називають *записом*. Таким чином, в окремій клітинці таблиці міститься інформація про значення одного параметра одного об'єкта (рис. 4.5).



**Рис. 4.5.** Відповідність між сутністю і таблицею:  
а — сутність у моделі «сутність-зв'язок»; б — таблиця в реляційній БД

Оскільки в кожному полі таблиці зберігаються значення того самого параметра різних об'єктів, всі ці значення мають однаковий *тип*, наприклад числовий, текстовий або дата/час. Інакше кажучи, тип має саме поле. Коли ви створюєте в базі даних таблицю, необхідно вказати назви й типи полів, вибрати ключові поля, а також задати назву самої таблиці. Крім того, для деяких полів варто задати додаткові параметри.

## Способи створення таблиць

### MS Access 2003

У СКБД Microsoft Access є три основних способи створення таблиць, кожному з яких відповідає своя команда на вкладці **Таблицы** головного вікна бази даних.

- ◆ **Создание таблицы в режиме конструктора** — введення назв полів, вибір їх типів та інших параметрів таблиці вручну.
- ◆ **Создание таблицы с помощью мастера** — визначення параметрів таблиці за допомогою діалогових вікон, де можна вибрати стандартні поля, перейменувати їх, задати назву таблиці, вказати ключові поля тощо.
- ◆ **Создание таблицы путем ввода данных** — автоматичне створення таблиці з 10 полями, що називаються **Поле 1**, **Поле 2** тощо; у таку таблицю можна відразу вводити дані. Поля можна буде згодом видалити, перейменувати, додати до них інші.

### MS Access 2007/2010

Існує чотири способи створення таблиць, кожному з яких відповідає своя кнопка на стрічці **Создание**.


- ◆ За допомогою кнопки **Таблица** створюють таблицю, де за умовчанням буде одне поле **Код**. Можна додати інші поля, але обирати їх тип та інші параметри незручно.
- ◆ Кнопку **Шаблони таблиц** призначено для створення таблиці стандартного типу, наприклад списку контактів.

- ◆ За допомогою кнопки **Списки SharePoint** таблиці автоматизовано створюються та наповнюються даними, отриманими з програми Microsoft SharePoint.
- ◆ Кнопка **Конструктор таблиц** дає змогу вводити назви полів, вибирати їхні типи та інші параметри вручну.

**Для допитливих.** У більшості СКБД немає жодного зі щойно згаданих способів створення таблиць. Замість них застосовують команду CREATE TABLE мови SQL. Нею можна скористатися і в СКБД Microsoft Access.

## Визначення назв і типів полів

Найуніверсальнішим та, мабуть, найзручнішим способом створення таблиці є використання конструктора, тому надалі ми розглядатимемо саме його. Вигляд *вікна конструктора таблиці* подібний до зображеного на рис. 4.6. У цьому вікні є три стовпці.

- ◆ У стовпці **Имя поля** слід увести назви полів (по одній у кожній клітинці).
- ◆ У стовпці **Тип данных** потрібно вибрати типи даних зі списків, що розкриваються кнопками  у правих частинах клітинок цього стовпця (див. поле *стать* на рис. 4.6). Щоб відобразити згадану кнопку, потрібно клацнути у клітинці.
- ◆ У стовпці **Описание** можна ввести коментар щодо призначення поля.

У Microsoft Access існує 10 стандартних типів даних. Призначення чотирьох із них — текстового, числового, грошового і дати/часу — цілком зрозуміле. Звернімо ще увагу на тип **Счетчик**, значеннями якого є цілі числа. У полях цього типу користувач не може вводити й змінювати дані, проте їх автоматично вводитиме СКБД під час створення нових записів. Фактично СКБД за допомогою полів-лічильників нумерує записи послідовними значеннями 1, 2, 3, ... . Поля-лічильники, як правило, є ключовими. Їх використовують для створення допоміжних кодів, що ідентифікують записи, у тому випадку, коли

значення коду не важливе, а важливо лише, щоб ці значення для різних записів були різними. Зокрема такий тип матиме поле код у таблиці Учні. Загалом у цій таблиці є п'ять полів: лічильник код, текстові поля прізвище, ім'я та стать, а також поле дата народження типу Дата/Час (рис. 4.6).

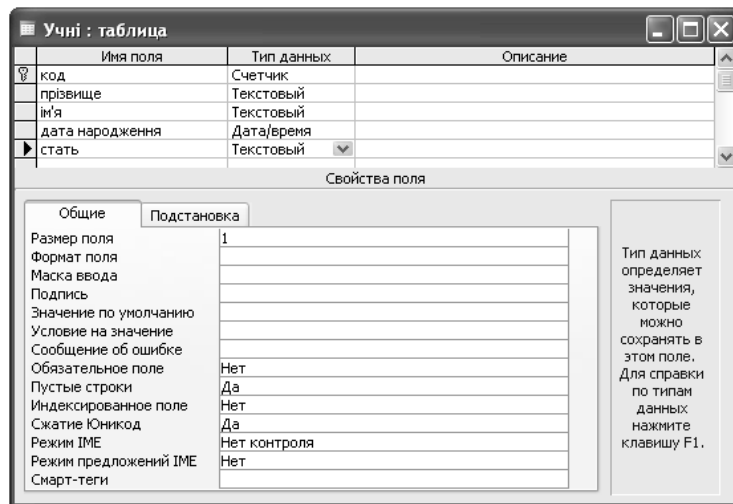


Рис. 4.6. Вікно конструктора таблиці



#### Для допитливих.

- ◆ Поле типу **МЕМО**, як і текстове поле, містить послідовність символів. Однак у поле **МЕМО** ви можете ввести до 65536 символів, у той час як у текстове поле — не більше 255.
- ◆ Тип **Логический** (Логічний) може мати два значення: **ИСТИНА** та **ХИБНИСТЬ**, що відображаються в таблиці як встановлений чи знятий прапорець. Цей тип може мати, наприклад, поле подружній стан (одружений — **ИСТИНА**, неодружений — **ХИБНИСТЬ**).
- ◆ Поле об'єкта **OLE** призначене для зберігання об'єктів найрізноманітніших типів: зображень, аудіо- та відеозаписів, форматowanego тексту, що можуть бути вбудованими в БД, а можуть зберігатися в окремих


## Визначення додаткових параметрів полів

Для кожного поля, крім його типу, можна задати й інші параметри, зокрема максимально допустиму кількість символів у рядку, який зберігається в текстовому полі. Для цього слід виділити поле (зліва від виділеного поля в Access 2003 відображається позначка ▶, а в Access 2007/2010 воно підсвічується жовтим) та визначити параметри в області **Свойства поля**, що розташована внизу вікна конструктора таблиці. Так, на рис. 4.6 видно, що для поля **стать** задано максимальну допустиму кількість символів 1 — це значення параметра **Размер поля** (Розмір поля), оскільки для збереження відомостей про **стать** достатньо одного символу: «ч» або «ж».

## Створення ключа таблиці

Коли ви визначите назви, типи та додаткові параметри полів таблиці, слід вказати, з яких полів складається її ключ. Для цього ключові поля потрібно виділити та клацнути кнопку  (Ключове поле) на панелі інструментів. Якщо ключове поле одне, то виділити його дуже просто: достатньо клацнути на ньому. Якщо ключ складається з кількох полів, то потрібно клацнути індикатор кожного з них, утримуючи клавішу **Ctrl**. Коли ви клацнете кнопку , зліва від виділених полів з'явиться позначка у вигляді ключа. На рис. 4.6 така позначка є біля поля **код** у таблиці **Учні**.

## Надання таблиці назви

Задавши ключові поля, закрийте вікно конструктора таблиці кнопкою . Буде виведено запит, чи потрібно зберегти зміни у структурі таблиці. Клацніть кнопку **Да**. Нарешті буде відображено вікно **Сохранение**, де слід ввести назву таблиці та клацнути кнопку **OK**.

## Операції з наявними таблицями

Значок новоствореної таблиці з'явиться на вкладці **Таблицы** головного вікна бази даних Access 2003 або області переходів

Access 2007/2010. Таблицю можна видалити, перейменувати, скопіювати, можна також змінити її структуру. Усі ці дії виконують за допомогою команд контекстного меню, яке відобразиться, якщо клацнути значок таблиці правою кнопкою миші (рис. 4.7).

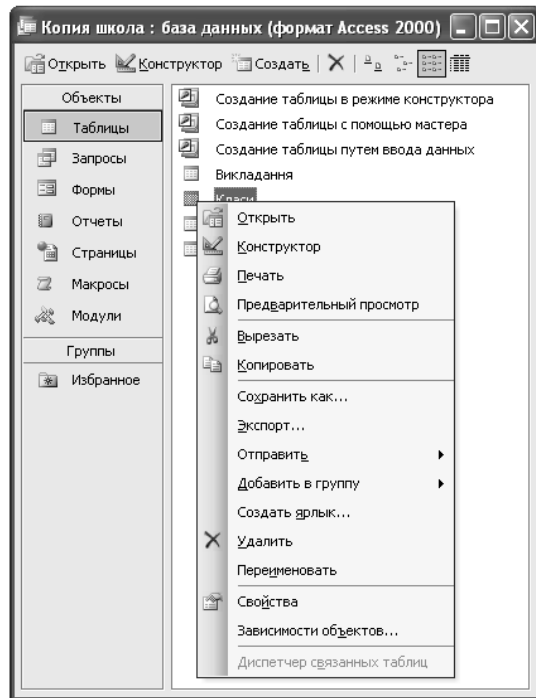


Рис. 4.7. Контекстне меню значка таблиці

- ◆ Щоб видалити таблицю, виберіть у контекстному меню її значка команду **Удалить**. Можна також виділити значок таблиці і клацнути кнопку **X** (Видалити).
- ◆ Для перейменування таблиці застосовують команду **Переименовать**.
- ◆ Щоб створити копію таблиці, в Access 2003 слід вибрати з контекстного меню її значка команду **Сохранить как** і у вік-

ні, що з'явиться, ввести ім'я копії таблиці. У Access 2007/2010 аналогічна команда — **Office ▶ Сохранить как ▶ Сохранить объект как**.

- ◆ Для змінення структури таблиці призначено команду **Конструктор**, що відкриває таблицю в уже знайомому вам вікні конструктора.

## Завдання 4.2

Створіть у базі даних **школа** таблиці **Учні**, **Класи** та **Учителі**, які відповідатимуть моделі «сутність-зв'язок», зображеній на рис. 2.8. Визначте самостійно, які типи повинні мати поля цих таблиць. Для поля **стать** задайте розмір 1 символ («ч» або «ж»).

## Завдання 4.3\*

Перейменуйте базу даних **школа** на **школи** та створіть у ній таблиці **Предмети**, **Школи**, **Директори** відповідно до моделі «сутність-зв'язок», зображеної на рис. 3.10. У таблиці **Директори** створіть тільки одне поле — **паспорт**, і зробіть його ключовим.

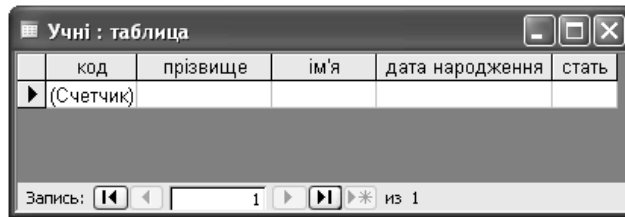
## Уведення та редагування даних

Досі ви виконували роль розробника бази даних, а тепер трохи попрацюєте як користувач. Нагадаємо, що розробник працює зі схемою бази даних, а користувач — із самими даними. Створення таблиць, визначення їх полів тощо — це був один з етапів розробки схеми БД. Зараз, коли в базі даних **школа** вже є три таблиці, можна ввести в кожен з них інформацію про кілька об'єктів.

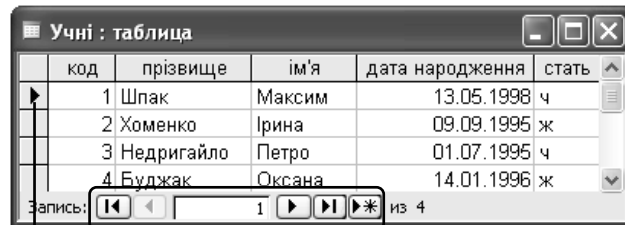
## Уведення даних

Щоб відкрити таблицю для введення даних, у головному вікні БД або в області переходів потрібно двічі клацнути її значок. Наприклад, клацнувши значок таблиці **Учні**, ви побачите таку таблицю, як на рис. 4.8, а. У цій таблиці є заголовок з назвами

полів та один порожній запис (нагадаємо, що так у БД називається рядок таблиці), у який ви можете ввести дані щодо одного учня. Коли ви почнете введення, з'явиться другий запис, коли почнете вводити дані в другий запис — з'явиться третій і т. д. На рис. 4.8, б зображено таблицю Учні з даними про чотирьох учнів.



а



індикатор запису

навігаційні елементи

б

Рис. 4.8. Введення даних у таблицю Учні: а — порожня таблиця; б — таблиця з інформацією про чотири об'єкти

Уведення даних у кожне поле варто завершувати натисканням клавіші **Enter** — тоді курсор буде переведено до наступного поля цього ж запису, а з останнього поля — до першого поля наступного запису.

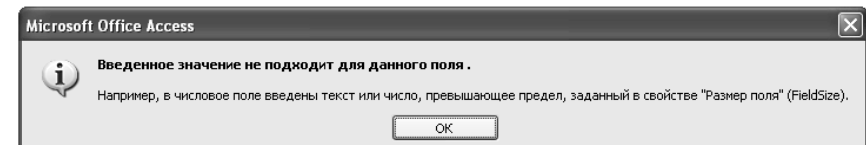
Зазначимо, що у вікні введення даних зліва від кожного запису розміщено його *індикатор* — квадрат, на якому можуть зображуватися різні позначки, а внизу вікна — *навігаційні еле-*

*менти*, за допомогою яких переміщуються записами (див. рис. 4.8, б).

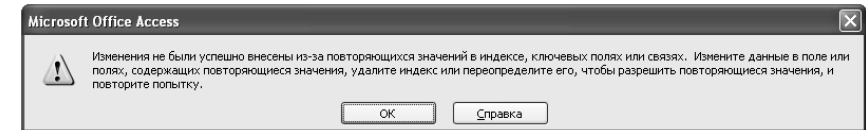
## Відповідність даних схемі БД

Як ми вже згадували в розділі 1, СКБД стежить за тим, щоб дані в базі узгоджувалися з її схемою. Зокрема, дані мають відповідати типам полів, тож якщо ви спробуєте ввести в поле дата народження замість дати слово, буде відображено повідомлення про помилку (рис. 4.9, а).

Ще одна вимога, за виконанням якої стежить СКБД, — унікальність значень ключа. Якщо ви не дотрималися цієї вимоги, наприклад, спробували ввести дані про двох вчителів з однаковим паспортом, буде відображено таке повідомлення про помилку, як на рис. 4.9, б.



а



б

Рис. 4.9. Повідомлення про помилку: а — невідповідність значення типу поля; б — повторення значень ключа

## Null-значення

Під час введення даних у таблицю деякі клітинки можна залишати порожніми. Потреба в цьому може виникнути з кількох причин, скажімо тому, що ви не знаєте, коли народився той чи інший учень і відкладаєте введення цієї інформації на

потім. Якщо клітинка порожня, то це означає, що значення певного параметра якогось об'єкта не існує. Така ситуація загрожує непередбачуваними наслідками: наприклад, неможливо сказати, істинним чи хибним буде вираз *кількість учнів > 20*, якщо значення *кількість учнів* неіснуюче. Щоб уникнути подібної непередбачуваності, СКБД у порожні клітинки записує спеціальне невідображуване значення *Null*, яке ще називають *порожнім значенням*. Ключові поля не можуть містити порожніх значень; якщо ви спробуєте залишити порожньою клітинку ключового поля, буде виведено таке повідомлення про помилку, як на рис. 4.10.

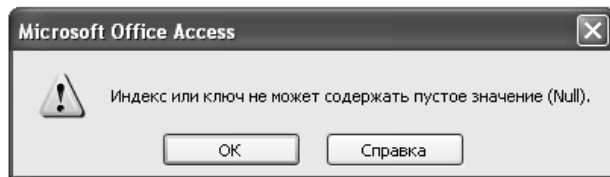


Рис. 4.10. Повідомлення про помилку, обумовлену введенням у ключове поле значення *Null*

## Переміщення таблицю

Коли таблицю відкрито в режимі введення даних, один із записів вважається *поточним*. У Access 2003 на індикаторі поточного запису є позначка **▶**, а в Access 2007/2010 індикатор виділено жовтим. В одному з полів поточного запису розташовано курсор — саме це поле доступне для редагування. Переміщуватися між записами та полями можна за допомогою клавіш керування курсором або миші.

Якщо записів у таблиці багато, для швидкого переходу між ними варто скористатися навігаційними елементами, що містяться внизу вікна введення даних (рис. 4.11):

- ◆ щоб перейти до наступного або попереднього запису, клацніть відповідно кнопку **▶** або **◀**;
- ◆ щоб зробити поточним перший або останній запис у таблиці, клацніть відповідно кнопку **◀◀** або **▶▶**;

- ◆ щоб перейти до запису з певним номером, введіть цей номер у текстове поле;
- ◆ для створення нового запису клацніть кнопку **▶\***.



Рис. 4.11. Навігаційні елементи

## Виділення фрагментів таблиць



У таблиці бази даних Microsoft Access можна виділяти фрагменти кількох типів:



- ◆ щоб виділити один запис, слід клацнути його індикатор;
- ◆ щоб виділити кілька суміжних записів, слід протягнути курсор миші по їхніх індикаторах;
- ◆ щоб виділити кілька несуміжних записів, потрібно по чергово клацнути їхні індикатори, утримуючи клавішу **Ctrl**;
- ◆ виділити частину значення поля можна встановивши на цьому значенні курсор та протягнувши його над потрібними символами;
- ◆ для виділення цілого значення поля, значень кількох полів одного або різних записів потрібно скористатися табличним курсором **+**, що відображається у разі підведення курсору до лівого верхнього кута клітинки.

## Редагування даних

Уведені в таблицю дані (цілі записи, значення окремих полів або фрагменти цих значень) можна вирізати за допомогою клавіш **Ctrl+X** або кнопки **✂** (Вирізати), копіювати в буфер обміну, використовуючи клавіші **Ctrl+C** або кнопку **📄** (Копіювати), та вставляти з нього за допомогою клавіш **Ctrl+V** або кнопки **📄** (Вставити), не забуваючи, звичайно, що будь-які значення мають відповідати типам полів, а значення ключових полів ще й не повинні повторюватися.

Значення поля можна відредагувати, клацнувши у відповідній клітинці. Зверніть увагу, що коли ви почнете вводити або редагувати дані, на індикаторі поточного запису буде відображено

значок . Це означає, що в записі є незбережені зміни. І тільки коли ви перейдете до іншого запису, введення поточного запису вважатиметься завершеним, зроблені вами зміни зберуться в базі даних, а індикатор поточного запису знову набуде вигляду . Такі обмеження цілісності, як унікальність значень ключа, перевіряються саме тоді, коли запис зберігається в базі даних, а не під час його редагування.

Цілі записи можна видаляти за допомогою команди **Удалить запис** контекстного меню індикатора запису. Крім того, в Access 2003 для цього призначено кнопку  (Видалити запис), а в Access 2007/2010 — кнопку  (Видалити) на стрічці **Главная**. Якщо ви хочете видалити відразу кілька записів, їх потрібно попередньо виділити, протягнувши курсор миші по індикаторах цих записів.

## Завдання 4.4

Уведіть у базу даних **школа** інформацію про такі об'єкти:


- ◆ класи 10А, 11А і 11Б;
- ◆ учителі:
  - Михайлюк Дмитро Семенович, чоловік, паспорт СН 410268, математик;
  - Сошко Катерина Миколаївна, жінка, паспорт СР 652320, біолог;
  - Корбут Василь Петрович, чоловік, паспорт СО 211517, математик;
  - Томчишин Віктор Георгійович, чоловік паспорт КН 200125, історик;
  - Петрова Ніна Володимирівна, жінка, паспорт СО 927453, фізик;
- ◆ учні:
  - Шпак Максим, 11.12.1996, хлопець;
  - Хоменко Ірина, 9.09.1995, дівчина;
  - Недригайло Петро, 1.07.1995, хлопець;
  - Буджак Оксана, 14.01.1996, дівчина.

## Завдання 4.5

Уведіть у базу даних **школи** інформацію про такі об'єкти:


- ◆ директор з паспортом СН 410268;
- ◆ школи:
  - Сомівська гімназія, розташована по вул. Гнатюка, 27;
  - ЗОШ №77, розташована по вул. Садовій, 14;
- ◆ предмети біологія, математика, хімія, фізика, українська мова.

## Збереження та відкриття бази даних

Можливо, ви помітили, що коли ви завершуєте введення даних у таблицю та закриваєте її вікно, кнопка  (Зберегти) стає неактивною. Це пояснюється тим, що база даних завжди перебуває у збереженому стані. Працюючи з БД, вам не потрібно періодично зберігати її, адже про це піклується сама СКБД.

Зазначимо, що команду **Файл** ▶ **Сохранить как** у Access 2003 призначено не для збереження бази даних, а для копіювання її таблиць та інших об'єктів.

**Для допитливих.** СКБД зберігає базу даних автоматично насамперед тому, що з нею можуть працювати одночасно кілька клієнтів (користувачів чи програм) і всі вони мають бачити БД в однаковому стані. А значить, не можна допускати ситуації, коли клієнт внесе зміни в базу, але не збереже їх: тоді він бачитиме одні дані, а інші клієнти — інші.

Відкривають базу даних в Microsoft Access так само, як і документи в інших офісних програмах: за допомогою кнопки  (Відкрити) або команди **Файл** ▶ **Открыть** у Access 2003 та команди **Открыть** кнопки Office в Access 2007/2010. Під час відкриття БД буде відображено одне або кілька вікон застережень з інформацією про те, що файл БД не є безпечним, може містити шкідливий код і т. п. В усіх таких вікнах потрібно



підтверджувати свій намір відкрити базу, клацаючи кнопки **Открыть**, **Да** або **ОК**.

## Висновки

- ◆ Кожній сутності в моделі «сутність-зв'язок» має відповідати таблиця в реляційній базі даних. Атрибутам сутності відповідають стовпці таблиці, які називають полями, а інформацію про кожен об'єкт сутності записують в окремому рядку таблиці, який в реляційних БД називають записом.
- ◆ Під час створення в базі даних таблиці необхідно вказати назви і типи полів, вибрати ключові поля, а також задати назву таблиці.
- ◆ Наявну в базі даних таблицю можна видалити, перейменувати, скопіювати, можна також змінити її структуру.
- ◆ Щоб увести в таблицю дані про набір об'єктів, у головному вікні БД потрібно двічі клацнути її значок та ввести інформацію про кожен об'єкт в окремому записі.
- ◆ СКБД забезпечує автоматичне збереження даних, що містяться в таблицях.

## Завдання для самостійного виконання

1. Створіть бази даних і таблиці в них відповідно до всіх моделей «сутність-зв'язок», які ви розробляли, виконуючи самостійні завдання до розділів 2 і 3.
2. Уведіть у кожну таблицю баз даних, створених у попередньому завданні, відомості про два–три об'єкти.

## Питання для роздумів

1. Якби в таблиці Класи не було визначено ключа, до яких негативних наслідків це могло б призвести?
- 2\*. Чому СКБД блокує спроби введення помилкових даних, а не відображає в клітинках коди помилок, як табличний процесор?

## Завдання для досліджень

1. Що станеться, коли в таблиці, у яку вже введено дані, змінити набір полів або їхні типи?
2. Навчіться створювати таблиці за допомогою майстра таблиць. Які переваги та недоліки має цей засіб порівняно з конструктором таблиць?
3. Створіть таблицю, у яку неможливо було б увести більше двох записів.
- 4\*. Чи можуть існувати:
  - ◆ однойменні поля в одній таблиці;
  - ◆ поля з однаковими назвами та типами даних в одній таблиці;
  - ◆ однойменні таблиці в одній БД?

Знайшовши відповіді на ці питання експериментально, спробуйте аргументувати, чому СКБД дозволяє чи не дозволяє створювати однойменні об'єкти в кожному з зазначених випадків.

## Розділ 5

# Створення зв'язків між таблицями

### Повторення

- ◆ Які різновиди зв'язків існують між сутностями предметної області?
- ◆ Як визначити різновид зв'язку?
- ◆ Що таке обмеження цілісності?
- ◆ Сформулюйте головний принцип семантичного моделювання.

Модель «сутність-зв'язок» складається з таких основних елементів, як сутності і зв'язки. Сутностям у базі даних відповідають таблиці, які ви навчилися створювати на попередньому уроці. Сьогодні ви дізнаєтеся, як створювати зв'язки між таблицями. Отже, опрацювавши матеріал цього розділу, ви вмітимете відображати в базі даних модель «сутність-зв'язок» у цілому.

### Створення зв'язку «один-до-багатьох»

Нагадаємо, що зв'язки між сутностями бувають трьох основних різновидів: «один-до-багатьох», «багато-до-багатьох» та «один-до-одного». Кожен із цих типів зв'язків у СКБД реалізують по своєму, і далі ми розглянемо, як саме. Почнемо із найпоширенішого типу зв'язку — «один-до-багатьох».

### Створення зв'язку між таблицями Учні та Класи

На минулому уроці в базі даних *школа* ви створили таблиці учнів, класів і вчителів. У таблицю класів ви ввели інформацію

цію про три класи (10А, 11А та 11Б), а в таблицю учнів — про чотирьох учнів. Але інформацію про те, який учень у якому класі вчиться, ви не вводили. Уважно подивившись на поля таблиць *Учні* та *Класи*, ви побачите, що таку інформацію просто немає куди вводити: поля *прізвище*, *ім'я* або *дата народження* таблиці *Учні* не призначено для зберігання цих даних, так само як і поле *назва* таблиці *Класи*. Як же нам вийти з цієї ситуації? Адже відомості про те, хто в якому класі вчиться, важливі і, очевидно, десь мають зберігатися. Теорія реляційних баз даних пропонує доволі просте рішення: у таблиці *Учні* потрібно створити додаткове поле *клас*, у яке і ввести назву класу кожного учня.

Припустимо, що наші учні вчаться у таких класах:

- ◆ Шпак Максим — у 10А;
- ◆ Хоменко Ірина — в 11А;
- ◆ Недригайло Петро — в 11Б;
- ◆ Буджак Оксана — в 11Б.

Після внесення необхідних змін таблиці *Учні* та *Класи* мають набути такого вигляду, як показано на рис. 5.1.

код	прізвище	ім'я	дата народження	стать	клас
1	Шпак	Максим	13.05.1998	ч	10А
2	Хоменко	Ірина	09.09.1995	ж	11А
3	Недригайло	Петро	01.07.1995	ч	11Б
4	Буджак	Оксана	14.01.1996	ж	11Б

назва
+ 10А
+ 11А
+ 11Б
*

Рис. 5.1. Таблиці *Класи* та *Учні* з інформацією про те, який учень у якому класі вчиться

Отже, поле *клас*, додане до таблиці *Учні*, стало тим засобом, за допомогою якого ми реалізували в базі даних зв'язок «учень вчиться у класі». Це зв'язок між сутностями *Учень* і *Клас*, що в моделі «сутність-зв'язок» виглядає так, як на рис. 5.2. Значення в доданому полі зберігають інформацію про зв'язки між об'єктами: конкретними учнями та класами. На

рис. 5.1 стрілками показано, з яким об'єктом Клас пов'язано кожен об'єкт Учень.

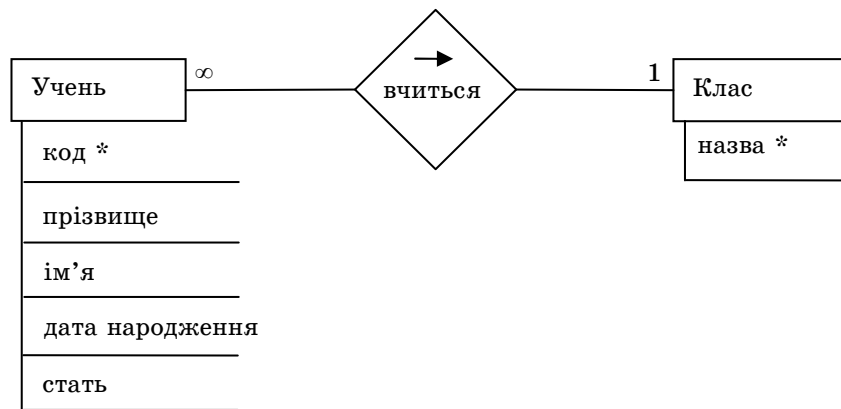


Рис. 5.2. Зв'язок між сутностями Учень і Клас у моделі «сутність-зв'язок»

Зауважте, що додаткові поля, призначені для збереження інформації про зв'язки, на моделі «сутність-зв'язок» не відображаються. Це пояснюється тим, що за допомогою додаткових полів зв'язки реалізують тільки в реляційних БД, а модель «сутність-зв'язок» від специфіки моделі даних абстрагована, вона може використовуватися в нереляційних БД і навіть не в базах даних взагалі.

**Для допитливих.** Збереження інформації про зв'язки об'єктів у додаткових полях є однією з визначальних рис, що відрізняють реляційні БД від усіх інших. У ієрархічних, мережових та об'єктно-орієнтованих БД об'єкт «учень» містив би вказівник на те місце в пам'яті, де зберігається об'єкт «клас», тобто зв'язки в цих БД реалізують *явно*. У реляційній БД об'єкт «учень» містить лише значення 10A, яке безпосередньо ні на що не вказує; щоб отримати відомості про такий клас, його потрібно ще знайти в таблиці Класи. Отже, в реляційних БД зв'язки між об'єктами реалізують *неявно*.

## Завдання 5.1

У таблиці Учні створіть поле клас того ж типу і з тими ж параметрами, що і поле назва в таблиці Класи, і введіть зазначені вище дані про навчання учнів у класах.

## Створення зв'язку «один-до-багатьох»: загальний випадок

Задамося питанням: чому для реалізації зв'язку «учень вчиться у класі» ми створювали додаткове поле клас у таблиці Учні, а не поле учень у таблиці Класи? Відповідь знайти не важко, якщо звернути увагу на тип цього зв'язку. Ми створили додаткове поле в таблиці Учні тому, що учень може вчитися тільки в одному класі, і не створювали його в таблиці Класи тому, що в класі може вчитися багато учнів. Справді, щоб зберегти в таблиці Класи відомості про навчання учнів у певному класі, довелося б для першого учня створювати додаткове поле учень1, для другого — учень2 і т. д. Неефективність такого підходу очевидна, а коли врахувати той факт, що кількість учнів у класах може бути різною, реалізувати подібне рішення виявиться взагалі неможливо.

Тепер припустимо, що таблиця Класи містить ще атрибут профіль (за його допомогою вказують, що клас філологічний, математичний тощо). Тоді які значення, назви чи профілі класів, потрібно вводити в поле клас таблиці Учні? Очевидно, що все одно назви, оскільки профіль не ідентифікує клас (наприклад, якщо вказати, що Шпак Максим вчиться у математичному класі, залишиться незрозумілим, у якому саме: десятому, одинадцятому чи якомусь іншому). Нагадаємо, що ідентифікувати об'єкти дозволяють значення ключових полів і саме тому, що поле назва є ключем у таблиці Класи, його значення потрібно вводити в додаткове поле клас таблиці Учні.

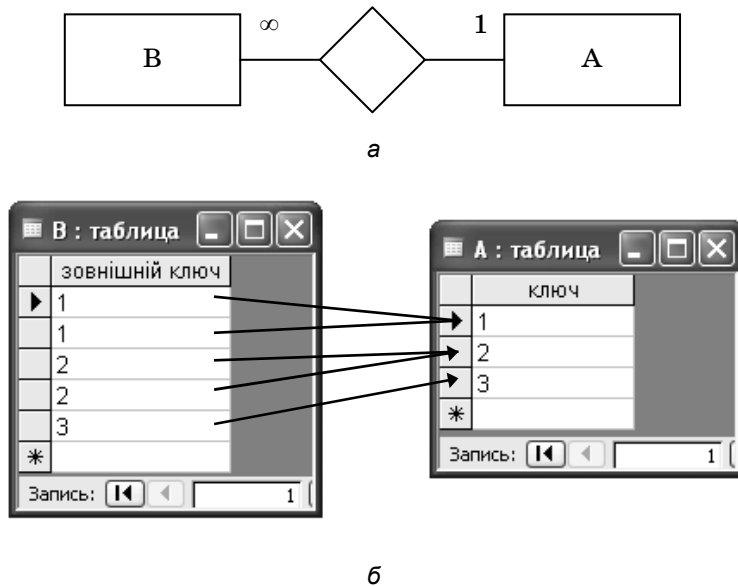
Поля, що містять значення ключів інших таблиць, називають *зовнішніми ключами*. Наприклад, поле клас — це зовнішній ключ таблиці Учні. Кажуть, що зовнішній ключ *посилається* на таблицю, значення ключа якої він містить. Так, зовнішній

ключ клас посилається на таблицю Класи. Ключ таблиці називають також *первинним ключем*, щоб підкреслити його відмінність від зовнішнього ключа.

**Зовнішній ключ** — це поле або набір полів, значення яких повинні належати множині значень первинного ключа деякої іншої таблиці.

Підсумовуючи сказане, сформулюємо правило моделювання зв'язку «один-до-багатьох» у реляційній базі даних.

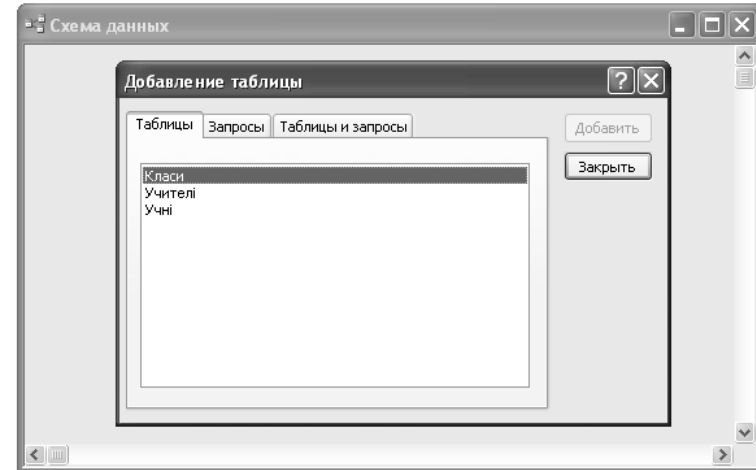
Щоб у реляційній базі даних реалізувати зв'язок «один-до-багатьох», зображений на рис. 5.3, а (кожному об'єкту А може відповідати багато об'єктів В, але кожному об'єкту В — лише один об'єкт А), у таблиці В потрібно створити зовнішній ключ, який посилається на таблицю А (рис. 5.3, б).



**Рис. 5.3.** Зв'язок «один-до-багатьох»: а — зображення в моделі «сутність-зв'язок»; б — реалізація в базі даних

## Графічне подання зв'язків у Microsoft Access

На панелі інструментів Microsoft Access 2003 та на стрічці **Работа с базами данных** Access 2007/2010 є кнопка (Схема даних), за допомогою якої схему бази даних можна зобразити графічно (якщо ви цієї кнопки в Access 2003 не бачите, зробіть активним головне вікно БД). Коли ви клацнете цю кнопку, буде відображено таке вікно, як на рис. 5.4.

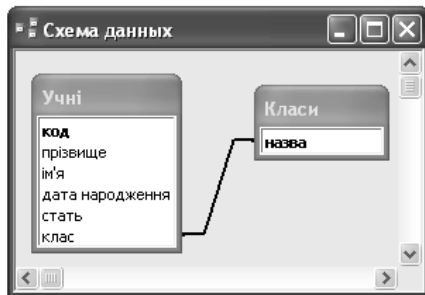


**Рис. 5.4.** Вікно **Схема данных** під час першого використання

У вікні **Добавление таблицы** потрібно, утримуючи клавішу **Ctrl** або **Shift**, виділити назви всіх таблиць і клацнути кнопку **Добавить**. У результаті всі таблиці буде відображено у вікні **Схема данных**. Вміст вікна можна редагувати: щоб видалити зображення таблиці, клацніть його правою кнопкою миші та виберіть у контекстному меню команду **Скрыть таблицу**, а щоб додати, скористайтеся кнопкою (Відобразити таблицю).

Зв'язки між таблицями у вікні **Схема данных** зображуються з'єднувальними лініями (рис. 5.5, а). Ці лінії з'єднують не просто зображення таблиць, а ті поля, за допомогою яких реалізується зв'язок. Як ви вже знаєте, зв'язок «один-до-багатьох» реалізують за допомогою ключового поля однієї таб-

лиці та зовнішнього ключа іншої. Саме ці поля потрібно з'єднати лінією у вікні **Схема даних**. Для цього слід клацнути ключове поле (назву такого поля записано жирним шрифтом) і, не відпускаючи лівої кнопки миші, перетягнути його на зовнішній ключ. Відкриється показане на рис. 5.5, б вікно **Изменение связей**, у якому достатньо лише клацнути кнопку **Создать** — і зв'язок буде відображено.



а

б

**Рис. 5.5.** Зв'язок «один-до багатьох»: а — позначення зв'язку на схемі даних; б — створення зв'язку у вікні **Изменение связей**

Проаналізуємо вміст вікна **Изменение связей** детальніше. У нижній його частині зазначено тип зв'язку (**один-ко-многим**), а

зверху містяться два списки: **Таблица/запрос** та **Связанная таблица/запрос**. У першому відображаються назви ключових полів таблиці, що перебуває на кінці зв'язку «один» (вона називається *головною*), а в другому — зовнішній ключ таблиці, що перебуває на кінці зв'язку «багато» (вона називається *зв'язаною*). За потреби поля, через які встановлюється зв'язок, можна змінити, клацнувши кнопку ▼ справа від назви ключового поля або зовнішнього ключа.

Параметри зв'язку можна змінити після його створення. Для цього слід двічі клацнути лінію зв'язку і задати необхідні параметри у вікні **Изменение связей**.

## Забезпечення цілісності даних

У новостворене поле клас таблиці *Учні* ми вводили назви класів з таблиці *Класи*, але СКБД дозволяє вводити в це поле майже будь-які дані, навіть назву неіснуючого у школі класу або, скажімо, слово *Привіт*. Звісно, такі дані будуть некоректними, адже учень не може вчитися в 21ш класі або у класі *Привіт*. Щоб гарантувати цілісність даних (згадайте розділ 1), СКБД має стежити за дотриманням простого правила: *У поле клас таблиці Учні неможна вводити значення, яких немає у полі назва таблиці Класи*.

Сформульоване правило є нічим іншим, як обмеженням цілісності: ми обмежуємо діапазон допустимих даних і тим самим гарантуємо їх цілісність. Неважко сформулювати це правило і в загальному вигляді:

*Зовнішній ключ зв'язаної таблиці не може містити значень, яких не містить ключ головної таблиці.*

Щоб примусити СКБД стежити за дотриманням цього обмеження цілісності, під час створення зв'язку у вікні **Изменение связей** (див. рис. 5.5, б) слід встановити прапорець **Обеспечение целостности данных**. Якщо ви це зробите, то спроба ввести в поле клас таблиці *Учні* значення на кшталт 21ш призведе до відображення повідомлення про помилку. На схемі даних у Microsoft Access лінії зв'язків з підтримкою цілісності вирізняються тим, що біля їхніх кінців зображуються символи мно-

жинності (1 або  $\infty$ ), так само, як на моделі «сутність-зв'язок» (рис. 5.6).



Рис. 5.6. Подання на схемі даних зв'язку з підтримкою цілісності

## Порушення обмежень цілісності

Підтримка цілісності, з одного боку, гарантує коректність даних, а з іншого, що більше в БД обмежень цілісності, то частіше виникають ситуації, коли зміни даних призводять до порушення цих обмежень. Можливі чотири випадки, у яких порушується обмеження цілісності, накладене зв'язком, і для кожного з них у СКБД передбачено механізми вирішення проблеми.

1. Коли створюють зв'язок, значення зовнішнього ключа вже порушують обмеження цілісності. У цьому випадку Microsoft Access не дозволить створити зв'язок, доки ви не усунете порушення.
2. Після створення зв'язку змінюють значення зовнішнього ключа зв'язаної таблиці. Якщо нового значення в ключі головної таблиці немає, СКБД не дозволить внести такі зміни, інакше зміни буде внесено і запис зв'язаної таблиці відповідатиме іншому запису головної таблиці.
3. Після створення зв'язку змінюють значення ключа головної таблиці. Зовнішній ключ зв'язаної таблиці після таких змін може порушувати обмеження цілісності. Microsoft Access дає змогу вирішити цю проблему двома шляхами:

- якщо під час створення зв'язку у вікні **Изменение связей** (див. рис. 5.5, б) не було встановлено прапорця **каскадное обновление связанных полей**, зміни буде заблоковано;
- якщо прапорець **каскадное обновление связанных полей** встановлено, СКБД здійснить *каскадне оновлення*: значення зовнішнього ключа буде змінено так само, як змінилися значення первинного ключа головної таблиці.

4. Після створення зв'язку запис головної таблиці видаляють. У цьому випадку може бути здійснено *каскадне видалення даних*: записи зв'язаної таблиці, що відповідають видаленому запису головної таблиці, також автоматично видалятимуться. Каскадне видалення здійснюється, якщо під час створення зв'язку у вікні **Изменение связей** було встановлено прапорець **каскадное удаление связанных полей**. Якщо прапорець знято, видалення блокуватиметься.

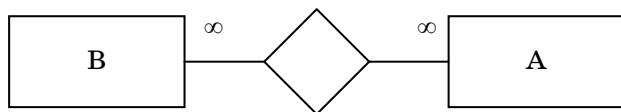
## Завдання 5.2

Відобразіть зв'язок між таблицями Класи та Учні у вікні **Схема даних** і забезпечте підтримку цілісності даних.

## Створення зв'язку «багато-до-багатьох»

Вище ми зазначали, що для моделювання зв'язку «учень вчиться у класі» зовнішній ключ неможна створювати в таблиці Класи, оскільки клас зв'язаний з багатьма учнями. Тож як бути у випадку зв'язку «багато-до-багатьох», наприклад зв'язку «учитель викладає в класі»? Адже подібні міркування приводять до висновку, що зовнішній ключ неможна створювати в жодній з двох таблиць-учасниць зв'язку. Насправді так і є: зв'язок «багато-до-багатьох» моделюють за допомогою додаткової таблиці, де і створюють зовнішні ключі.

*Щоб у реляційній базі даних змоделювати зв'язок «багато-до-багатьох» між таблицями А і В (рис. 5.7, а), потрібно створити додаткову таблицю С, а в ній — два зовнішні ключі, що посилаються на таблиці А і В (рис. 5.7, б).*



а

б

**Рис. 5.7.** Зв'язок «багато-до-багатьох»: а — зображення в моделі «сутність-зв'язок»; б — реалізація в базі даних

Наприклад, для моделювання зв'язку «вчитель викладає в класі» ми створимо таблицю Викладання з двома полями: учитель і клас. Поле учитель буде зовнішнім ключем, що посилається на таблицю Учителі, а поле клас — зовнішнім ключем, що посилається на таблицю Класи.

Для зберігання інформації про те, що вчитель Сошко Катерина Миколаївна викладає у 10А та 11А класах, у таблиці викладання необхідно створити два записи: в одному зазначити номер паспорта цієї вчительки (СР 652320) та назву класу 10А, а в іншому — той самий номер паспорта та назву класу 11А (рис. 5.8, а).

У такий спосіб у таблицю Викладання можна ввести інформацію про викладання одного вчителя в кількох класах. Нічого нам не завадить ввести в цю таблицю й інформацію про викладання в одному класі кількох учителів. Наприклад, на рис. 5.8, б зображено відомості про те, що в 10А класі викладають вчителі з паспортами СР 652320 та СО 927453. Таким чином, допоміжна таблиця дозволила нам зв'язати учителя з

багатьма класами, а клас — з багатьма вчителями. У підсумку це і є зв'язком «багато-до-багатьох» між таблицями Учителі та Класи.

а

б

**Рис. 5.8.** Моделювання зв'язку вчителя з класами

**Для допитливих.** У таблиці Викладання, як і в будь-якій іншій, варто створити первинний ключ. Поле учитель не є ключем, оскільки його значення можуть повторюватися. Те саме можна сказати і про поле клас. Але пара значень атрибутів учитель, клас повторюватися не повинна. Отже, у таблиці Викладання ключ складається з двох полів. Створення цього ключа гарантуватиме, що інформація про викладання вчителя X у класі Y зберігатиметься лише в одному записі, і це узгоджується з головним принципом семантичного моделювання, який ми сформулювали в розділі 2. Нагадаємо, що для створення ключа з кількох полів потрібно у вікні конструктора таблиці виділити їх усі, утримуючи клавішу **Ctrl**, після чого клацнути кнопку .

Щоб зобразити зв'язок «учитель викладає у класі» графічно, у вікні **Схема даних** необхідно з'єднати ключі таблиць Учителі та Класи із відповідними зовнішніми ключами таблиці Викладання. Варто також подбати про забезпечення цілісності даних для цих зв'язків. У результаті ви маєте отримати схему, показану на рис. 5.9.



Рис. 5.9. Зв'язок «багато-до-багатьох» на схемі даних

Як видно з рис. 5.9, зв'язок «багато-до-багатьох» фактично являє собою два зв'язки «один-до-багатьох», у яких з боку «багато» розташовано допоміжну таблицю. Це справедливо не лише для зв'язку «учитель викладає у класі», а й для будь-якого зв'язку типу «багато-до-багатьох». Отже, сформульоване вище правило побудови зв'язку «багато-до-багатьох» можна перефразувати так.

*Щоб створити між таблицями А і В зв'язок «багато-до-багатьох», необхідно створити допоміжну таблицю С і приєднати до неї таблиці А і В зв'язками «один-до-багатьох».*

### Завдання 5.3

Промодельуйте в базі даних **школа** зв'язок «учитель викладає в класі»: створіть таблицю **Викладання** з зовнішніми ключами **учитель** і **клас**, типи яких мають бути такими ж, як типи первинних ключів таблиць **Учителі** і **Класи**. Уведіть такі дані:

- ◆ Сошко Катерина Миколаївна викладає у 10А та 11А класах;
- ◆ Михайлюк Дмитро Семенович викладає в 11Б класі;

- ◆ Корбут Василь Петрович викладає в 11А і 11Б класах;
- ◆ Томчишин Віктор Георгійович викладає в 11А класі;
- ◆ Петрова Ніна Володимирівна викладає в 10А, 11А і 11Б класах.

Номери паспортів учителів не вводьте з клавіатури, а копіюйте з таблиці **Учителі**.

Зобразіть зв'язок у вікні **Схема даних** та забезпечте підтримку цілісності даних.

### Створення зв'язку «один-до-одного»

Зв'язок «один-до-одного», так само як і зв'язок «один-до-багатьох», створюють із використанням зовнішнього ключа. Але у зв'язку «один-до-одного» зовнішній ключ має специфічну властивість: його значення не повинні повторюватися. Щоб пояснити цю тезу, звернімося знову до рис. 5.1, де проілюстровано зв'язок «один-до-багатьох». Один запис таблиці **Класи** міг бути зв'язаний з багатьма записами таблиці **Учні** саме тому, що в таблиці **Учні** значення зовнішнього ключа могли повторюватися (наприклад, значення 11Б на рис. 5.1 у таблиці **Учні** ми бачимо двічі). Якщо заборонити такі повтори, то кожен клас буде зв'язаний не більше, ніж з одним учнем, тобто ми реалізуємо зв'язок «один-до-одного».

Існує два способи гарантувати зазначену властивість зовнішнього ключа.

1. Вважати, що первинний ключ є також зовнішнім. Нагадаємо, що неповторення значень — це основна властивість первинного ключа, і якщо його вважати зовнішнім, отримаємо саме те, що потрібно для зв'язку «один-до-одного».
2. Створити для зовнішнього ключа додаткове поле і зробити його *індексованим без повтorenь*. Індекссовані поля — це ті, за значеннями яких СКБД шукає записи в таблиці. Щоб СКБД могла відрізнити один запис від іншого, розробнику надано можливість заборонити повторення значень індексованого поля — саме цим ми і скористаємося.



Для допитливих. Індексовані поля, або поля з *індексами*, — один із найважливіших механізмів реляційних СКБД, який дозволяє кардинально прискорити пошук даних. Індекс — це відсортована послідовність значень одного або кількох полів. Якби індексів не було, то для пошуку заданого значення у певному полі системі довелося б перевіряти всю таблицю, що нерідко містить мільйони записів. Але відсортованість індексу дозволяє знаходити задане значення серед мільйона інших усього за допомогою 20 операцій порівняння.

Під час побудови зв'язку «один-до-багатьох», нагадаємо, зовнішній ключ ми створювали в тій таблиці, що розташована з боку «багато». Коли ж ідеться про зв'язок «один-до-одного», зовнішній ключ можна створювати в будь-якій таблиці.

Розглянемо зв'язок «учитель є класним керівником» і припустимо, що зовнішній ключ міститиметься в таблиці Класи. Визначимо, у який спосіб його краще створити: вважати зовнішнім первинний ключ (поле назва) чи створити для нього додаткове поле. Якщо обрати перший спосіб, то значення з поля назва мають міститися і в полі паспорт таблиці Учителі, що суперечить здоровому глузду. Отже, зовнішній ключ може розміщуватися лише в додатковому полі.

Якщо зовнішній ключ створювати в таблиці Учителі, висновок буде той самий: поле паспорт не може містити назв класів, а отже, бути зовнішнім ключем.

Таким чином, для зовнішнього ключа в одній із таблиць необхідно створити додаткове поле. Нехай це буде таблиця Класи, а поле називатиметься класний керівник. Коли ви створюватимете це поле в режимі конструктора таблиці, в області властивостей поля зі списку **Индексированное поле** слід вибрати елемент **Да (совпадения не допускаются)** — саме це дозволить створити зв'язок типу «один-до-одного» (рис. 5.10).

Коли в Microsoft Access на схемі даних відображають зв'язок «один-до-одного», первинний ключ потрібно перетягувати на зовнішній, а не навпаки, оскільки СКБД вважатиме зв'язаною

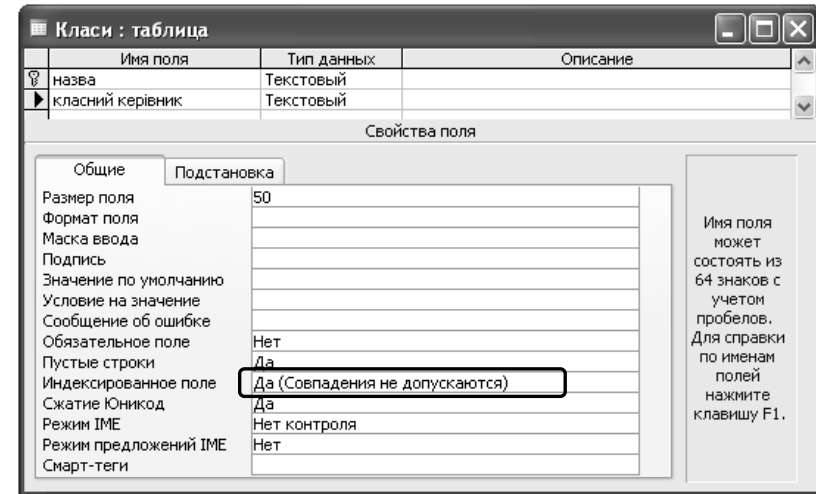


Рис. 5.10. Визначення параметрів зовнішнього ключа для зв'язку «один-до-одного»

ту таблицю, на яку перетягують поле, а головною — ту, з якої тягнуть. Так, вам потрібно перетягнути поле паспорт таблиці Учителі на поле класний керівник таблиці Класи. Якщо встановити прапорець **Обеспечение целостности данных**, то ви побачите біля кінців зв'язку символи «1» і Microsoft Access підтримуватиме обмеження цілісності, тобто не дозволить вводити в поле класний керівник таблиці Класи значення, які не є номерами паспортів учителів, представлених у таблиці Учителі. Загалом схема даних набуде такого вигляду, як на рис. 5.11.

## Завдання 5.4

Промодельуйте в базі даних **школа** зв'язок «учитель є класним керівником», створивши у таблиці Класи зовнішній ключ класний керівник та зробивши його індексованим полем без повторень. Уведіть такі дані:

- ◆ Сошко Катерина Миколаївна — класний керівник 10А класу;

- ◆ Петрова Ніна Володимирівна — класний керівник 11А класу;
- ◆ Корбут Василь Петрович — класний керівник 11Б класу.

Зобразіть зазначений зв'язок у вікні **Схема даних** та забезпечте підтримку цілісності даних.



Рис. 5.11. Схема бази даних школа

## Реалізація концепцій поглибленого семантичного моделювання\*

У розділі 3 ми розглянули такі концепції, як обов'язковість зв'язків, слабкі сутності, зв'язки «є», зв'язки між кількома сутностями та зв'язки сутностей самих із собою. Для їх реалізації у СКБД не потрібно ніяких додаткових засобів, крім тих, які ми вже вивчили, однак спосіб їх застосування має певні особливості.

### Обов'язковість зв'язків\*

Нагадаємо, що зв'язок називається обов'язковим з боку певної сутності, якщо всі об'єкти цієї сутності повинні брати участь у зв'язку. Наприклад, учень повинен вчитися у деякому класі, а клас повинен мати класного керівника (див. рис. 3.2).

Здамося питанням: коли запис не бере участі у зв'язку, яким буде значення зовнішнього ключа в цьому записі? Очевидно, що воно має бути порожнім. Отже, щоб забезпечити обов'язковість участі записів певної таблиці у зв'язку, слід гарантувати, що її зовнішній ключ не міститиме Null-значень. Для цього параметру **Обязательное поле** зовнішнього ключа слід надати значення **Да** (вікно параметрів показано на рис. 5.10).

Для допитливих. Якщо зв'язок «один-до-одного» обов'язковий з боку якоїсь таблиці, то саме в ній потрібно створювати зовнішній ключ. Наприклад, щоб участь записів таблиці Клас у зв'язку «учитель є класним керівником» була обов'язковою, у цій таблиці слід створити зовнішній ключ класний керівник і зробити це поле обов'язковим — тоді його значення не зможуть бути порожніми, а отже, кожен клас повинен мати класного керівника. Якби ми створювали зовнішній ключ у таблиці Учителі, то обов'язковою могли б зробити тільки участь у зв'язку вчителів, що було б нелогічним, адже не кожен учитель повинен бути класним керівником.

Зв'язки «один-до-багатьох», за допомогою яких моделюють зв'язок «багато-до-багатьох», варто робити обов'язковими з боку допоміжної таблиці, оскільки кожен її запис обов'язково відповідає двом записам основних таблиць.

### Завдання 5.5\*

У базі даних **школи** забезпечте обов'язковість зв'язку «учень вчиться у класі» з боку таблиці Учні та зв'язків «вчитель є класним керівником» і «школа містить клас» з боку таблиці Класи.

### Слабкі сутності\*

Як зазначалося в розділі 3, слабкою називається сутність, частина ключа якої складається з атрибутів іншої сутності. Ці атрибути повторюватимуться в обох сутностях і, як неважко

помітити, становитимуть зовнішній ключ підтримувального зв'язку у слабкій сутності. Отже, слабка сутність має таку властивість, що частина її ключа є зовнішнім ключем. Так, на рис. 5.12, а зображено слабку сутність Класи та підтримувальний зв'язок її з сутністю Школи, а на рис. 5.12, б — реалізація цієї конструкції в схемі даних. Ключ слабкої сутності — це пара атрибутів назва (класу) та назва школи, а зовнішній ключ — атрибут назва школи.

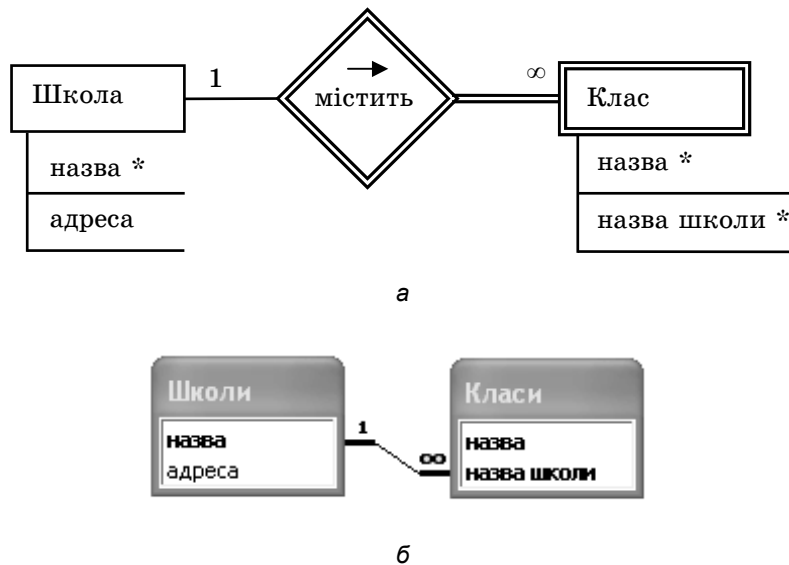


Рис. 5.12. Слабкі сутності: а — на моделі «сутність-зв'язок»; б — на схемі даних

### Завдання 5.6\*

У базі даних **школи** створіть такий зв'язок між таблицями Класи та Школи, як показано на рис. 5.12, б. Для кожного класу вкажіть, що він належить одній з тих шкіл, записи яких ви створювали у завданні 4.5. Крім того створіть другий 10А клас, який має належати іншій школі, ніж уже створений 10А.

### Зв'язок «є»\*

Зв'язок «є», нагадаємо, створюють між сутністю-загальним типом і сутністю-різновидом. Наприклад, директор — це різновид учителя, а учитель — більш загальний, ніж директор, тип педагогічного працівника (рис. 5.13, а). Цей зв'язок завжди має множинність «один-до-одного» і обов'язковий з боку сутності-різновиду.

Основна перевага використання зв'язків «є» полягає в тому, що сутність-різновид автоматично «успадковує» всі атрибути сутності-загального типу і їх не потрібно вказувати в сутності-різновиді. У таблиці, що представляє сутність-різновид, достатньо створити лише ключ, який повторюватиме ключ таблиці, що представляє сутність-загальний тип (рис. 5.13, б). Наприклад, учитель Михайлюк Дмитро Семенович є директором. У таблиці Директори ми записуємо тільки номер його паспорта, а в таблиці Учителі — повну інформацію. Фактично за допомогою таблиці Директори ми лише позначаємо деяких учителів: «він є директором».

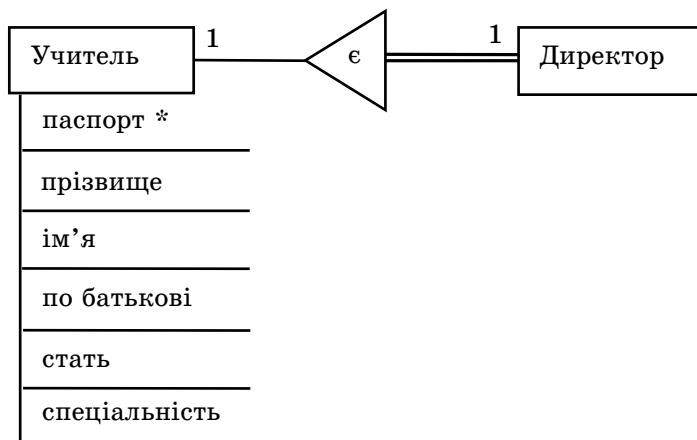
Очевидно, що для реалізації зв'язку «один-до-одного» типу «є» потрібно вважати первинний ключ таблиці-різновиду зовнішнім ключем. А отже, моделюючи цей зв'язок в MS Access, на схемі даних слід перетягнути ключ таблиці-загального типу (наприклад, таблиці Учителі) на ключ таблиці-різновиду (наприклад, таблиці Директори). На схемі даних зв'язок «є» виглядатиме, як показано на рис. 5.13, в.

### Завдання 5.7\*

У базі даних **школи** створіть зв'язок між таблицями Директори та Учителі, а також Директори і Школи (див. рис. 3.11). Вкажіть, що Михайлюк Дмитро Семенович є директором Сомівської гімназії, а Томчишин Віктор Георгійович — директором ЗОШ №77.

### Зв'язок між кількома сутностями\*

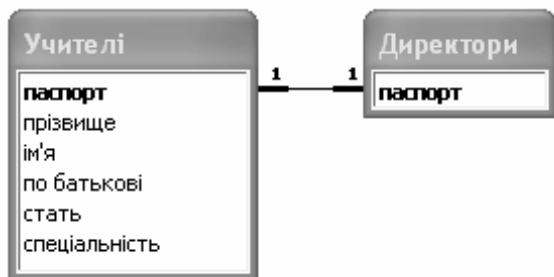
Зв'язок між кількома сутностями завжди моделюють за допомогою допоміжної таблиці незалежно від того, яку він має



а

паспорт	прізвище	ім'я	по батькові	стать	спеціальність
CH 410268	Михайлюк	Дмитро	Семенович	ч	математик
CO 211517	Корбут	Василь	Петрович	ч	математик
CP 652320	Сошко	Катерина	Миколаївна	ж	біолог
KN 200125	Томчишин	Віктор	Георгійович	ч	історик
CO 927453	Петрова	Ніна	Володимирівна	ж	фізик

б



в

Рис. 5.13. Зв'язок «є»: а — у моделі «сутність-зв'язок»; б — промодельований даними; в — на схемі даних

множинність. Як і для зв'язку «багато-до-багатьох», у цій таблиці створюють зовнішні ключі і з'єднують її бінарними зв'язками з таблицями, що беруть участь у зв'язку (рис. 5.14).

Якщо множинність тернарного зв'язку з усіх боків становить «багато», то первинний ключ допоміжної таблиці складатиметься з усіх її зовнішніх ключів, а бінарні зв'язки, що з'єднують її з основними таблицями, матимуть множинність «один-до-багатьох» (рис. 5.14). Усі інші типи тернарних зв'язків і зв'язків вищих степенів ми не розглядатимемо, однак ви можете поміркувати над їх моделюванням самостійно (див. питання для роздумів 5).



Рис. 5.14. Тернарний зв'язок

### Завдання 5.8\*

У базі даних школи створіть тернарний зв'язок між таблицями Учителі, Предмети та Класи згідно з рис. 5.14. Уведіть інформацію про те, що кожен учитель викладає той предмет, з якого він має спеціальність, у класах, зазначених у завданні 5.3. Крім того, вкажіть, що К.М. Сошко викладає в 11А класі хімію.

### Зв'язок сутності самої з собою\*

Є кілька способів моделювання зв'язку сутності самої з собою у СКБД, більш чи менш вдалих з огляду на те, скільки обмежень цілісності вони дозволяють підтримувати. У розділі 3 ми побудували зв'язок «шлюб», який зображено на рис. 5.15, а. З ним

пов'язані такі обмеження цілісності, як множинність «один-до-одного» та різна стать учителів, що беруть участь у зв'язку.

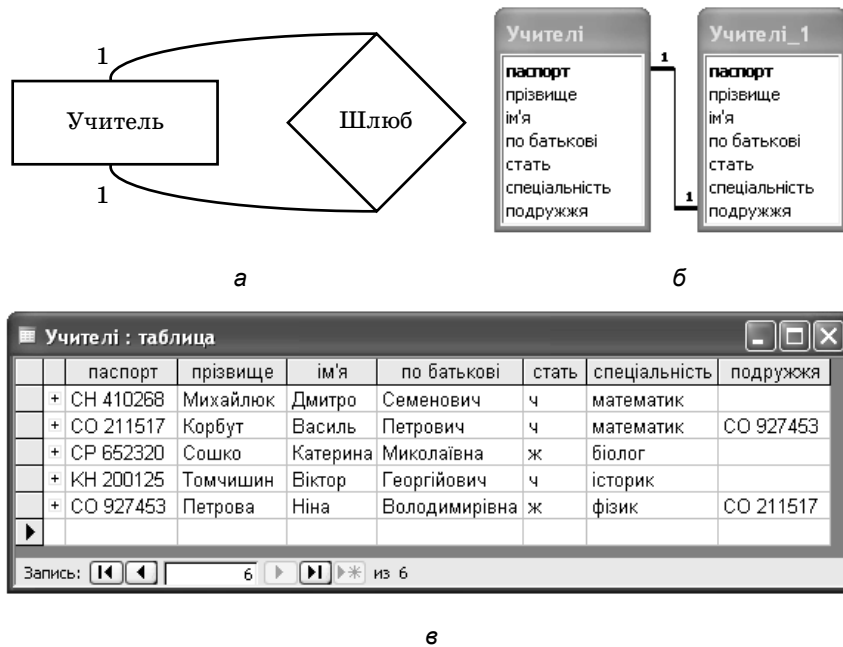


Рис. 5.15. Зв'язок сутності самої з собою: а — у моделі «сутність-зв'язок»; б — на схемі даних; в — зв'язок між об'єктами

Найпростіший спосіб його реалізації — ще раз додати таблицю Учителі до схеми даних (її другий екземпляр буде автоматично названо Учителі\_1) та зв'язати два екземпляри цієї таблиці зв'язком «один-до-одного». У таблиці Учителі можна створити зовнішній ключ, назвемо його подружжя, зробити його індексованим полем без повторень та зв'язати з первинним ключем іншого екземпляра цієї ж таблиці (рис. 5.15, б). Зауважте, що Учителі\_1 — це не нова таблиця, а просто назва другого подання таблиці Учителі на схемі даних.

Щоб показати, скажімо, що Петрова Ніна Володимирівна є дружиною Корбута Василя Петровича, слід у поле подружжя

запису Н.В. Петрової ввести номер паспорта В.П. Корбута, а поле подружжя В.П. Корбута — номер паспорта Н.В. Петрової (рис. 5.15, в).

Розглянутий спосіб моделювання зв'язку «шлюб» не гарантує, однак, що стать чоловіка і жінки буде різною. Щоб забезпечити дотримання цього обмеження цілісності, можна створити дві таблиці: Учителі-жінки та Учителі-чоловіки, з'єднати їх одне з одним та з таблицею учителів (рис. 5.16). Зв'язки між таблицями Учителі-жінки і Учителі, а також Учителі-чоловіки і Учителі будуть зв'язками «є»: учитель-жінка є учителем, так само, як і учитель-чоловік. Зауважте, що поле стать за такої реалізації буде зайвим, адже стать учителя визначатиметься тим, у якій таблиці вказано його паспорт: чоловіків чи жінок.



Рис. 5.16. Зв'язок «шлюб», промодельований шляхом створення двох нових сутностей

Для допитливих. Другий підхід до моделювання зв'язку «шлюб» теж має недоліки, адже, наприклад, ніщо не забороняє ввести одного того самого вчителя і до таблиці чоловіків, і до таблиці жінок водночас. Щоби повністю адекватно змодельовати зв'язок сутності самої з собою, слід скористатися засобами, вивчення яких виходить за межі шкільного курсу інформатики.

## Завдання 5.9\*

1. У базі даних **школи** створіть зв'язок «шлюб» між учителем та учителем шляхом додавання другого екземпляра таблиці Учителі до схеми даних. Вкажіть, що Корбут Василь Петрович одружений на Петровій Ніні Володимирівні.
2. Створіть копію бази даних **школи**, а в ній — зв'язок «шлюб» між учителем та учителем шляхом створення таблиць Учителі-жінки та Учителі-чоловіки згідно з рис. 5.16. Вкажіть, що Корбут Василь Петрович одружений на Петровій Ніні Володимирівні.

## Висновки

- ◆ Зовнішній ключ — це поле або набір полів, значення яких повинні належати множині значень первинного ключа деякої іншої таблиці. Кажуть, що зовнішній ключ посилається на таблицю, значення ключа якої він містить. Зовнішні ключі використовують для моделювання зв'язків між таблицями та окремими об'єктами.
- ◆ Якщо певні дві таблиці з'єднані зв'язком, то та, яка містить зовнішній ключ, називається зв'язаною, а інша — головною.
- ◆ Щоб реалізувати зв'язок «один-до-багатьох» (кожному об'єкту А може відповідати багато об'єктів В, але кожному об'єкту В — лише один об'єкт А), потрібно в таблиці В створити зовнішній ключ, що посилається на таблицю А.
- ◆ Щоб змоделювати між таблицями А і В зв'язок «багато-до-багатьох», потрібно створити додаткову таблицю С, а в ній — два зовнішні ключі, що посилаються на таблиці А і В. Інакше кажучи, до таблиці С слід приєднати таблиці А і В зв'язками «один-до-багатьох».
- ◆ Щоб змоделювати між таблицями А і В зв'язок «один-до-одного», потрібно в одній з таблиць створити зовнішній ключ, що посилається на іншу таблицю. Цей зовнішній ключ має також бути первинним ключем або індексованим полем, у якому повторення значень не допускається.

## Завдання для самостійного виконання

У базах даних, створених вами у завданні для самостійного виконання з розділу 4, реалізуйте зв'язки між таблицями відповідно до моделей «сутність-зв'язок», спроектованих у розділах 2 і 3. Уведіть дані про зв'язки між кількома об'єктами в кожній БД.

## Питання для роздумів

1. Припустимо, що в базі даних **школа** містяться відомості про 20 учнів 11А класу і 20 учнів 11Б класу, але назви класів переплутали, і всіх учнів 11А класу записали в 11Б, а учнів 11Б класу — в 11А. Вкажіть, як виправити помилку, змінивши дані в записах лише тричі. Вважатимемо, що даних про вчителів у базі немає.
- 2\*. Опишіть алгоритм, який дозволить за допомогою 20 операцій порівняння знайти запис із заданим значенням індексованого поля у таблиці, що містить 1 000 000 записів. Нагадаємо, що індекс — це впорядкована послідовність значень поля.
- 3\*. Зв'язки «один-до-одного» майже ніколи не роблять обов'язковими з обох боків. Поясніть, чому саме. Які негаразди спричинив би такий зв'язок?
- 4\*. Наведіть приклад двох сутностей з будь-якої предметної області, між якими існує зв'язок «один-до-одного», що реалізується за допомогою з'єднання первинних ключів, але це не зв'язок «є».
- 5\*. Припустимо, що в один день трамвайний вагон курсує одним маршрутом, але в різні дні може курсувати різними. Тоді між сутностями Трамвайний вагон, День і Маршрут існує тернарний зв'язок. Визначте множинність цього зв'язку і промоделюйте його в базі даних MS Access на схемі даних та за допомогою самих даних.

## Розділ 6

# Інтерфейс користувача бази даних

### Повторення

- ◆ Як у базах даних Microsoft Access моделюють зв'язок «багато-до-багатьох» між двома сутностями?
- ◆ Якщо між сутностями існує зв'язок «один-до-багатьох», то як моделюють зв'язок між об'єктами цих сутностей?
- ◆ Чим відрізняється конструктор таблиці від майстра створення таблиці?
- ◆ Яка таблиця називається головною, а яка підлеглою?

Дуже часто ми використовуємо бази даних, навіть не підозрюючи про це. Наприклад, коли ми входимо до облікового запису електронної пошти, поштова служба звертається до розміщеної на сервері БД, щоб перевірити, чи збігається збережений в базі даних пароль з тим, який було введено в екранну форму. Однак жодних об'єктів бази даних — полів, записів, таблиць і т. п. — під час цього ми не побачимо, оскільки вони «приховані» за інтерфейсом прикладної програми, у нашому випадку — служби веб-пошти. Такий підхід виправданий і зрозумілий: можна тільки уявити, наскільки незручно було б користувачеві самому шукати потрібну таблицю в БД, у ній — «свій» запис тощо. Насправді користувач майже ніколи не оперує з базою даних напряму, а послуговується натомість інтерфейсом прикладних програм, де він може вводити дані у спеціально сконструйованих діалогових вікнах. У деякі СКБД, і зокрема у Micro-

soft Access, вбудовано засоби розробки інтерфейсу користувача для введення, редагування та виведення даних з бази. Сьогодні ми познайомимося з одним із таких засобів, *формами* — діалоговими вікнами, призначеними для введення і редагування даних у таблицях.

### Створення форм у режимі майстра

У MS Access 2003 для створення форм, так само, як і для створення інших об'єктів, є два автоматизовані засоби: *майстер* та *конструктор*. Можливості майстра вужчі, але використовувати його легше, тому насамперед навчимося працювати з цим засобом. У MS Access 2007/2010 в області **Форми** на стрічці **Создание** є ще кілька кнопок, за допомогою яких створюють прості стандартні форми, але спосіб їх використання очевидний.

### Створення форми для введення даних в одну таблицю

Найчастіше форму створюють для введення та редагування даних в одній таблиці. Щоб створити таку форму за допомогою майстра, у Access 2003 потрібно в головному вікні бази даних у меню об'єктів вибрати вкладку **Форми**, а потім двічі клацнути команду **Создание форми с помощью мастера**, а в Access 2007/2010 — виконати команду **Мастер форм** з меню кнопки **Другие формы** в області **Формы** на стрічці **Создание**.

У першому вікні майстра (рис. 6.1) зі списку **Таблицы и запросы** слід вибрати таблицю, для якої створюється форма, і всі поля таблиці буде відображено у списку **Доступные поля**. Ті поля, у які вводитиме дані користувач, необхідно перенести до списку **Выбранные поля** за допомогою кнопок **>** (перенести виділене поле) та **>>** (перенести всі поля). Якщо будуть вибрані зайві поля, для скасування вибору слід скористатися кнопкою **<** чи **<<**.

На рис. 6.1 показано, як створюється форма для таблиці **Учні**. Спочатку необхідно вибрати всі поля, крім поля **код**, оскільки його тип — лічильник і дані в нього користувач не вводитиме. Найлегше це зробити, вибравши спочатку всі по-

ля за допомогою кнопки >>, а потім скасувавши вибір поля кодом кнопкою <.

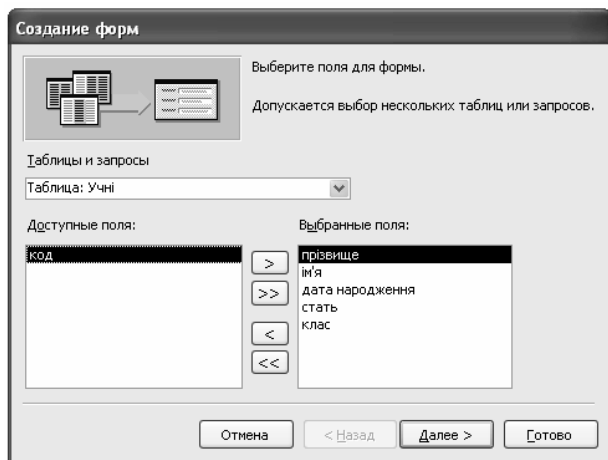


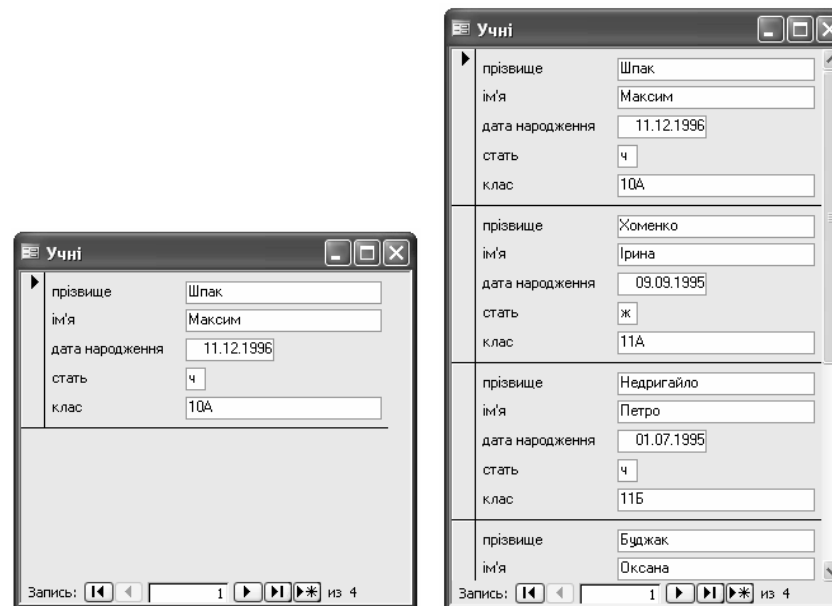
Рис. 6.1. Перше вікно майстра створення форми

Далі, у другому вікні майстра потрібно вказати спосіб подання інформації у формі. Найбільш поширеними вважають три способи.

- ◆ **В один стовбець** (в один стовпець) — у вікні форми відобразатимуться дані лише одного запису (рис. 6.2, а). Щоб перейти до наступного запису, слід скористатися навігаційними елементами, розташованими внизу вікна форми і детально описаними в розділі 3 (див. рис. 3.10).
- ◆ **Ленточний** (стрічковий) — у вікні форми відобразатимуться дані кількох записів (рис. 6.2, б). Для переходу до записів, які не вмістилися у вікно форми, використовують навігаційні елементи.
- ◆ **Табличний** (табличний) — форма нічим не відрізнятиметься від самої таблиці, за винятком того, що у ній можуть бути відображені не всі поля, а лише деякі (рис. 6.2, в).

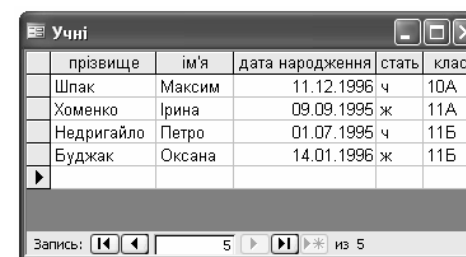
У третьому вікні майстра вибирають дизайн, тобто колірну гаму, шрифти й інші параметри самої форми та розміщених на

ній елементів керування. У четвертому вікні слід увести ім'я форми (воно може збігатися з іменем таблиці) та клацнути кнопку **Готово**.



а

б



в

Рис. 6.2. Способи відображення інформації у формі: а — в один стовпець; б — стрічковий; в — табличний



Найпростіший спосіб створення форми для однієї таблиці — виділити таблицю в області переходів і клацнути кнопку **Форма, Разделенная форма** або **Несколько элементов** на стрічці **Создание**. Відразу після цього буде відображено макет форми, що матиме вигляд, як на одному з рис. 6.2, *а–в*, залежно від того, яку кнопку ви клацнули. Цей макет можна відредагувати візуальними засобами, а потім закрити та використовувати саму форму.

Створена форма відображується на вкладці **Формы** головного вікна бази даних у MS Access 2003 і в області переходів MS Access 2007/2010. Двічі клацніть її, і форма відобразиться у вигляді, що залежить від обраного вами способу подання даних. Ви можете вводити дані в поля форми редагувати їх, переміщуватися між полями (за допомогою миші або клавіші **Tab**), та записами (з використанням навігаційних елементів), видалити записи кнопкою **✕**, додавати їх кнопкою **➤** або **➤\*** тощо.

## Завдання 6.1

Створіть форми для таблиць *Учні* та *Класи*. За допомогою форми створіть запис 10Б класу та введіть відомості про учнів Григорука Петра, 5.05.1997, та Райчук Олену, 13.01.1998, що навчаються в цьому класі. Відкривши таблиці, переконайтеся, що дані справді додано.

## Створення форми для введення даних у кілька таблиць

Інколи зручно і логічно, щоб з однієї форми дані вводилися в кілька таблиць. Наприклад, коли вводимо відомості про вчителя, було б доречно ввести відразу й інформацію про те, у яких класах він викладає, тобто передбачити на одній формі поля для введення даних у таблиці *Учителі* та *Викладання*. Готова форма в цьому випадку може виглядати так, як показано на рис. 6.3. Фактично в одному вікні ми бачимо дві форми: *голов-*

*ну*, де введено дані про вчителя, та *підлеглу*, де вказують класи, у яких цей учитель викладає. Кожна з цих форм пов'язана зі своєю таблицею: *головна* — з таблицею *Учителі*, а *підлегла* — з таблицею *Викладання*. Крім того, форми зв'язані одна з одною: у *підлеглу* форму вводять не довільні класи, а класи саме того вчителя, інформація про якого зберігається в *головній* формі.

Як видно з рис. 6.3, у кожній із цих форм (*головній* та *підлеглій*) є свої навігаційні елементи. Ті, що розташовані у нижній частині *головної* форми, дають змогу переходити від учителя до учителя, а ті, що у *підлеглій*, — переходити між класами, де викладає відображений у *головній* формі вчитель.

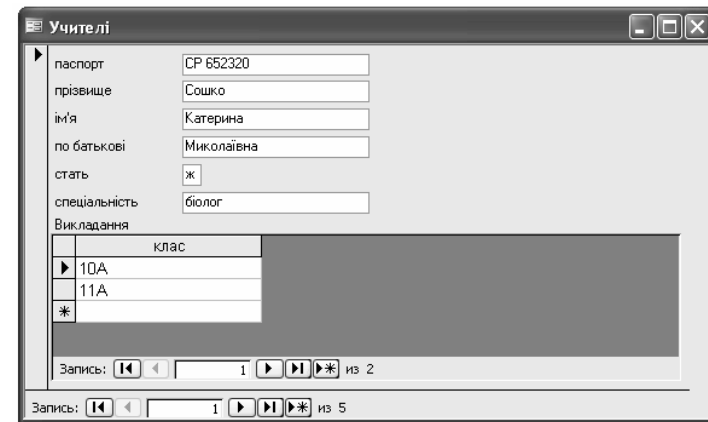


Рис. 6.3. Форма для введення даних у таблиці *Учителі* та *Викладання*

Коли ви створюєте форму для кількох таблиць, перший крок роботи з майстром буде майже таким самим, як у разі створення форми для однієї таблиці. Відмінність лише в тому, що зі списку **Таблицы и запросы** потрібно спочатку вибрати одну таблицю, відібрати її поля, а потім повторити ті самі дії для інших таблиць. Якщо ви створюєте форму для таблиць *Учителі* та *Викладання*, перше вікно майстра форм слід заповнити так, як показано на рис. 6.4.

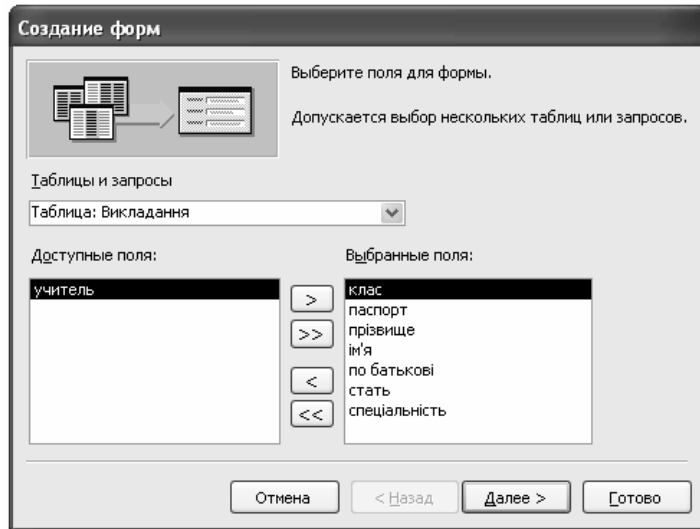


Рис. 6.4. Перше вікно майстра створення форми для таблиць Учителі та Викладання

Вибрано всі поля таблиці Учителі та поле клас таблиці Викладання. Це означає, що в одному вікні форми буде вводиться вся інформація про вчителя, а також назви класів, у яких він викладає. Зауважте, що поле учитель таблиці Викладання, де зберігаються номери паспортів, не вибрано. Це пояснюється тим, що, увівши значення в поле паспорт таблиці Учителі, ми вже знатимемо, про викладання якого вчителя йдеться, тож і вводити номер його паспорта ще кілька разів немає потреби.

На другому кроці майстра створення форми буде відображено вікно, подане на рис. 6.5. У ньому ви маєте вказати, якій із двох таблиць відповідатиме головна форма, вибравши цю таблицю зі списку в лівій частині вікна. У нашому випадку головна форма будуватиметься за таблицею Учителі. Перемикач внизу вікна визначає спосіб з'єднання головної форми з підлеглою: якщо обрано положення **Подчиненные формы**, то підлегла форма відобразатиметься всередині головної, інакше

вона відкриватиметься кнопкою на головній формі і називатиметься *зв'язаною*.

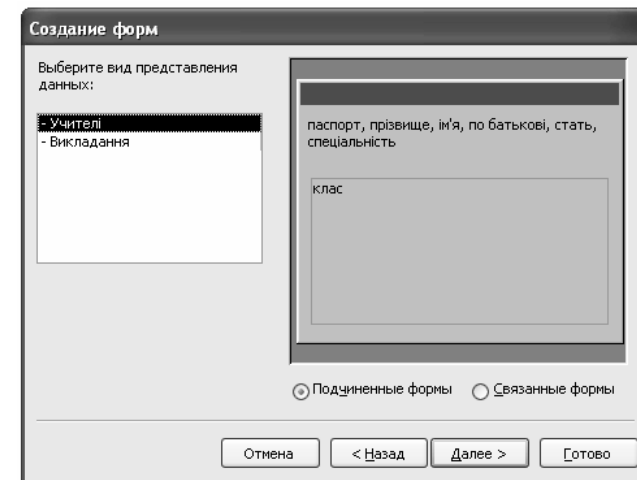


Рис. 6.5. Друге вікно майстра створення форми для двох таблиць

Третє і четверте вікна майстра будуть такими самими, як і в разі створення форми для однієї таблиці.

## Завдання 6.2

Створіть форму для таблиці Учителі із вкладеною формою для таблиці Викладання. За допомогою форми додайте інформацію про вчителя Савчука Василя Семеновича (паспорт СН 512840, чоловік, математик), вказавши, що він викладає у 10А та 10Б класах.

## Яку форму варто робити головною

Задасмося питанням: чому головною ми зробили форму саме для таблиці Учителі, а підлеглою — для таблиці Викладання, а не навпаки? Відповідь знайти нескладно, якщо згадати тип зв'язку між цими таблицями: одному запису в таблиці Учителі

відповідає багато записів у таблиці Викладання, а отже, вибравши учителя, зручно переглянути всі класи, учнів яких він навчає. Якби ми в головній формі відобразили таблицю Викладання, кожному її запису відповідав би один учитель, для якого створювати підлеглу форму було б недоцільно.

Наведені вище міркування можна узагальнити: *якщо таблиці з'єднані зв'язком «один-до-багатьох», форму таблиці, розташованої з боку «багато» можна зробити підлеглою до форми таблиці, розташованої з боку «один».*

## Редагування форм у конструкторі

З точки зору розробника інтерфейсу БД форма являє собою набір елементів керування — текстових полів, написів, списків і т.п. Їх можна переміщувати, розтягувати, стискати, додавати, видаляти, змінювати розмір і тип шрифтів тощо. Ці дії виконують у *конструкторі форм*. Щоб відкрити його, потрібно з контекстного меню значка форми в головному вікні бази даних або в області переходів вибрати команду **Конструктор**. Вікно конструктора форми зображено на рис. 6.6.

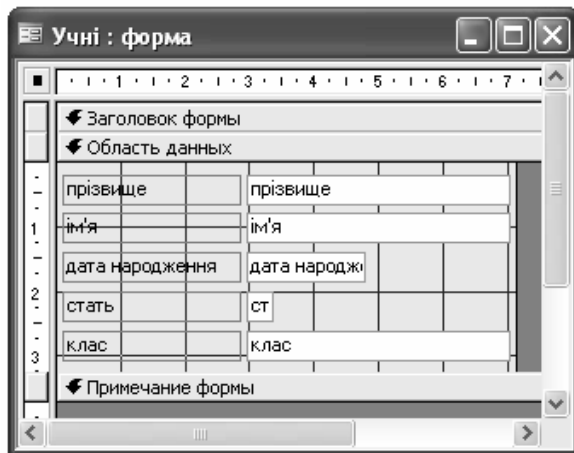


Рис. 6.6. Вікно конструктора форми

**Для допитливих.** Вікно конструктора форми поділено на три області.



- ◆ В області даних відображаються дані з таблиці БД. Вміст цієї області змінюється, коли ви переходите від одного запису до іншого.
- ◆ Область заголовка розташовано над областю даних, а область приміток — під нею.

Який би запис не відображався у формі, вміст цих областей залишається незмінним.

За умовчанням області заголовка та приміток згорнуті: є смужки з назвами цих областей, але самих областей не видно. Щоб розгорнути область, потрібно встановити курсор на нижню межу смужки та протягнути її вниз.

### MS Access 2003

Елементи керування в конструкторі форми виділяють, клацаючи їх мишею або обводячи рамкою виділення. Навколо виділеного елемента з'являються маркери, перетягуючи які можна змінювати розміри елемента. Коли виділено кілька елементів, їх можна автоматично вирівняти за допомогою команди **Вирівнять** контекстного меню.

Майстер форм для кожного поля таблиці БД створює два елементи керування: поле для введення тексту та напис із назвою поля таблиці. Ці елементи згруповані: коли виділити один, автоматично виділиться й інший. Якщо ви встановите курсор на згрупованих елементах ближче до нижньої межі, він набуде вигляду руки: . Захопивши цей курсор, ви зможете переміщувати елементи разом (рис. 6.7). Щоб перемістити окремий елемент, слід скористатися курсором , який з'являється у разі встановлення вказівника миші на лівий верхній кут елемента.

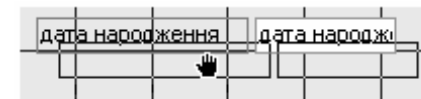


Рис. 6.7. Переміщення елементів керування формою

Ще одна важлива дія — додавання до форми нового елемента керування. Це можна робити за допомогою команди **Вставка** ▶ **Елемент ActiveX** або панелі елементів, де зібрано найважливіші елементи керування (рис. 6.8).



Рис. 6.8. Панель елементів керування

### MS Access 2007/2010

Коли ви працюєте з конструктором форми, з'являються стрічки **Конструктор** та **Упорядочить**. За допомогою стрічки **Конструктор** ви можете додавати до форми нові елементи керування, налаштовувати параметри шрифтів та ін., а за допомогою стрічки **Упорядочить** керують способом розташування елементів на формі.

Елементи керування на формі можуть бути розташовані згідно з певним *макетом*, який обирають в області **Макет елемента управління** на стрічці **Упорядочить**. Власне, є два макети:

- ◆ **Табличний** — подібний до електронної таблиці, де підписи розташовано в області заголовка, а дані — у стовпцях під підписами;
- ◆ **В столбик** — поля розташовано одне під одним, а підписи — зліва від них.

Майстер розміщує елементи керування на формі згідно з макетом **В столбик**. Щоб елементи керування вільно переміщувати по формі та змінювати їх розміри, макет слід видалити, клацнувши кнопку **Удалить** в області **Макет елемента управління**. Якщо ви цього не зробите, елементи керування переміщуватимуться тільки всі разом, а їх розміри змінюватимуться лише пропорційно.

Якщо макет видалено, то, виділивши кілька елементів керування, з ними можна виконати ті самі дії, що і в будь-якому векторному графічному редакторі: вирівняти, перемістити на задній чи передній план, згрупувати, прив'язати до сітки тощо. Кнопки для цих дій розташовано на стрічці **Упорядочить**.

Майже всі дії з елементом керування можна виконати за допомогою його контекстного меню. У результаті вибору команди **Свойства** цього меню відображається вікно властивостей, де можна ввести значення всіх параметрів у текстовому вигляді. Щоб відкрити вікно властивостей всієї форми, слід клацнути правою кнопкою миші позначку  у лівому верхньому куті форми та вибрати з меню команду **Свойства**.

Ми не розглядатимемо параметрів елементів керування детально, а звернемо увагу лише на найважливіший з них. У вікні властивостей більшості елементів керування є вкладка **Данные**, а на ній — однойменний параметр. Значенням цього параметра є назва поля таблиці БД, з яким *зв'язано* елемент керування. Термін «зв'язано» тут означає, що дані, введені в елемент керування, автоматично записуватимуться у відповідний елемент таблиці БД. Клацнувши кнопку  справа від значення цього параметра, елемент керування можна зв'язати з іншим полем.


Щоб дізнатися, як використовувати конструктор форм, виконайте вправу 6.1.

### Вправа 6.1

Змініть форму **Учні**, забезпечивши вибір дати народження учня з календаря.

1. Клацніть правою кнопкою миші форму **Учні** в області переходів або на вкладці **Формы** головного вікна бази даних і виберіть з контекстного меню команду **Конструктор**.
2. У вікні конструктора форми виділіть текстове поле і напишіть дату народження та видаліть їх, натиснувши клавішу **Del**.
3. Додайте до форми елемент керування **Календар**. Для цього у Access 2003 виконайте команду **Вставка** ▶ **Елемент ActiveX**,

а в Access 2007/2010 клацніть кнопку **Элементы ActiveX** на стрічці **Конструктор**. У вікні, що відкриється, виберіть пункт **Елемент керування Календар 11.0, Календар 12.0** або подібний (ці елементи розташовано ближче до кінця списку). Календар з'явиться у формі.

4. Скоріш за все, календар затулятиме інші елементи керування. Щоб уникнути цього, розширте область даних форми, перетягнувши вниз верхню межу смужки **Примечание формы**, та перемістіть календар на вільне місце. Можливо, потрібно буде перемістити й інші елементи керування.
5. Над календарем створіть напис дата народження. Для цього на панелі елементів керування або на стрічці **Конструктор** клацніть кнопку **Az** (Напис), а потім клацніть форму там, де напис розміщуватиметься, і введіть текст.
6. Зв'яжіть календар із полем дата народження. Для цього клацніть календар правою кнопкою миші, виберіть у контекстному меню команду **Свойства**, на вкладці **Данные** клацніть поле справа від параметра **Данные**. З'явиться кнопка . Клацніть її і виберіть у списку, що відкриється, поле дата народження. Закрийте вікно властивостей.
7. Закрийте вікно конструктора форми, зберігши внесені зміни. Готову форму показано на рис. 6.9.
8. Відкрийте форму і змініть дату народження одного з учнів, скориставшись календарем. Після цього відкрийте таблицю **Учні** і переконайтеся, що дату народження дійсно змінено.

## Висновки

- ◆ Форма — це діалогове вікно, призначене для введення даних у таблиці, а також для їх редагування.
- ◆ Як правило, для обробки даних в одній таблиці створюють одну форму.
- ◆ Майстер форм дає змогу створити форму з текстовими полями для введення даних у поля таблиці.
- ◆ Конструктор форм дозволяє відредагувати набір елементів керування у формі та їхні параметри.

- ◆ Якщо таблиці з'єднано зв'язком «один-до-багатьох», форму таблиці, розташованої з боку «багато», можна зробити підлеглою до форми таблиці, розташованої з боку «один».



Рис. 6.9. Форма **Учні** з календарем

## Завдання для самостійного виконання

Створіть форми для введення даних у бази, розроблені у завданнях для самостійного виконання з попередніх розділів.

## Завдання для досліджень

1. Навчіться відображати малюнки як тло форми.
2. Дізнайтеся, як додавати підлеглу форму до вже існуючої, і додайте підлеглу форму **Викладання** до форми **Класи**.
- 3\*. Зробіть так, щоб спроби ввести в поле **стать** у формі **Учні** будь-який символ, крім «ч» або «ж», блокувалися.

## Розділ 7°

# Автоматизація роботи з базою даних

### Повторення

- ◆ Що таке зовнішній ключ?
- ◆ Для чого призначено форми в базах даних MS Access?
- ◆ Які є способи подання інформації у формах?
- ◆ Назвіть сім найпоширеніших елементів керування.

Створюючи форми, ви мали помітити, що на них можна розміщувати не менше двох десятків різноманітних елементів керування. Однак ви працювали лише з трьома: текстовим полем, написом і календарем. У цьому розділі ми опишемо, як використовувати ще два важливих елементи: поля зі списками та кнопки. Перші застосовують переважно для введення інформації про зв'язки між записами, а другі — для навігації базою даних, тобто для переходу до форм, таблиць та інших об'єктів.

### Інтерфейс для введення інформації про зв'язки

В інтерфейсі, розробленому нами в попередньому розділі, є незручність, з якою ви зіткнетесь під час введення інформації про класного керівника. Припустимо, вам потрібно вказати, що Томчишин Віктор Георгійович є класним керівником 10Б класу.

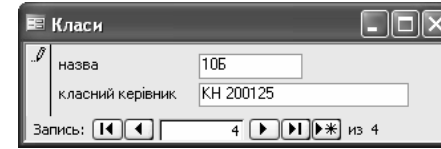



Рис. 7.1. Форма Класи без розкритого списку

Для цього у формі Класи (рис. 7.1) вам необхідно знайти запис 10Б класу і ввести в поле класний керівник номер паспорту цього вчителя. Але, звичайно, користувач бази даних номерів паспортів учителів не пам'ятає,

а отже, йому потрібно відкрити форму або таблицю Учителі, знайти вчителя Томчишина, скопіювати номер його паспорта, повернутися до форми Класи і вставити цей номер в поле класний керівник. Погодьтеся, процедура незручна і навряд чи дружне до користувача програмне забезпечення змушуватиме його виконувати ці дії. У вдало спроектованому інтерфейсі замість текстового поля класний керівник на формі Класи мав би бути розкритий список (його називають ще полем зі списком), з якого користувач міг би вибрати потрібного вчителя. Як перетворювати текстові поля на списки, описано у праві 7.1.

### Вправа 7.1

Забезпечте можливість вибору класного керівника на формі Класи зі списку всіх учителів.

1. Відкрийте в режимі конструктора форму Класи, яку ви створювали в завданні 6.1. Якщо цієї форми в базі даних школа немає, створіть її за допомогою майстра.
2. Клацніть правою кнопкою миші текстове поле класний керівник і виберіть з контекстного меню команду **Преобразовать элемент в > Поле со списком**.
3. Перейдіть у вікно властивостей поля зі списком класний керівник, клацнувши його правою кнопкою миші та вибравши з контекстного меню команду **Свойства**.
4. На вкладці **Данные** вікна властивостей поля зі списком встановіть курсор у поле **Источник строк** і виберіть з меню кнопки , що відобразиться справа, таблицю Учителі. У такий спосіб ви вкажете, що елементи розкритого списку мають формуватися з записів таблиці Учителі.

5. Закрийте вікно властивостей, а потім — вікно конструктора форми, зберігши при цьому зміни. Відкрийте форму як користувач та перевірте, як працює розкритий список.

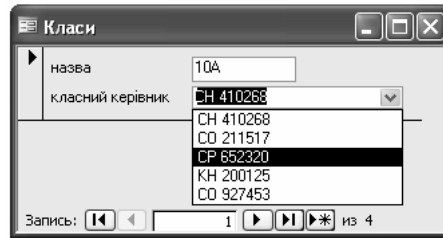


Рис. 7.2. Форма Класи з розкритим списком

- Якщо ви все правильно зробили, у ньому відобразатиметься список паспортів учителів (рис. 7.2). Це також незручно, оскільки ви не пам'ятаєте, який паспорт кому належить. Отже, потрібно зробити так, щоб у списку відображались принаймні номери паспортів, прізвища та імена вчителів.
6. Поверніться до конструктора форми та знову відкрийте вікно властивостей поля зі списком. Перейдіть на вкладку **Макет** і в поле **Число стовбців** уведіть значення 3. Це означає, що в розкритому списку відобразатимуться значення з трьох перших стовпців таблиці-джерела, тобто таблиці *Учителі*.
7. Щоб у списку відобразилися значення з трьох стовпців, він має бути широким. Тому поверніться у вікно конструктора форми та розширте область даних, пересунувши вправо її межу. Потім збільште ширину самого списку.
8. Відкрийте форму *Класи* як користувач. Тепер розкритий список матиме приблизно такий вигляд, як на рис. 7.3. Перейдіть до запису 10Б класу і виберіть зі списку класного керівника — Томчишина Віктора.

Підб'ємо підсумки: ми створили розкритий список для введення значень у поле класний керівник — зовнішній ключ таблиці *Класи*. Елементами цього списку є значення первинного ключа паспорт таблиці *Учителі*, на яку посилається згаданий зовнішній ключ, а також ще двох полів. Неважко описати і загальний випадок, коли у формах доцільно створювати розкриті списки.

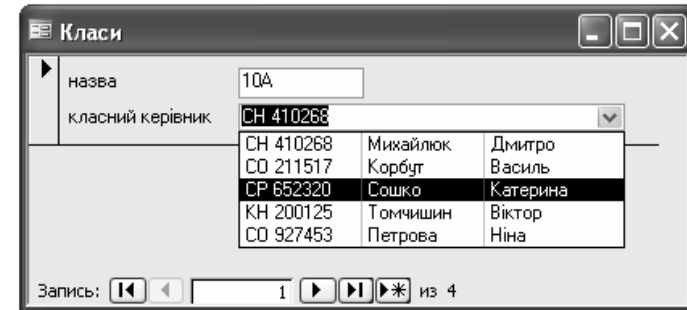


Рис. 7.3. Розкритий список зі значеннями з трьох стовпців

Розкриті списки створюють для введення значень зовнішніх ключів. Елементами розкритого списку є значення первинного ключа тієї таблиці, на яку посилається зовнішній ключ і, можливо, ще деяких її полів.

Якщо керуватись цим правилом, розкриті списки у БД *школа* варто створити ще для вибору класу у формі *Учні*, а також класу і вчителя у формі *Викладання*. Однак це не настільки важливо, адже назву класу легко ввести і в текстове поле, а форму *Викладання* ми зробили підлеглою, і вчитель під час роботи з нею визначається автоматично.

## Завдання 7.1

Забезпечте можливість вибрати клас на формі *Учні* зі списку всіх класів.

**Для допитливих.** Можна визначати не лише кількість полів таблиці, що відобразатимуться у списку, але й їхню ширину, перелічуючи відповідні значення через крапку з комою в полі **Ширина стовбців** на вкладці **Макет** вікна властивостей поля зі списком. Наприклад, якщо в поле **Число стовбців** ввести число 5, а в поле **Ширина стовбців** — значення 0 см; 2 см; 0 см; 0 см; 2 см, то відобразатимуться тільки другий і п'ятий стовпці, кожен 2 см завширшки.

## Навігація базою даних

Виконавши вправи і завдання цього й попереднього розділу, ви створили достатньо зручні форми для введення даних у базу **школа**. Однак щоб відкрити ці форми, потрібно використовувати головне вікно бази даних або область переходів, у яких міститься багато інших об'єктів і тому знайти потрібну форму не так просто. Щоб завершити проектування дружнього користувачеві інтерфейсу бази даних, потрібно ще створити меню або інші засоби переходу між об'єктами бази. Інтерфейс БД має бути таким, щоб усі дії з базою користувач міг виконувати через нього, не взаємодіючи безпосередньо з СКБД.

Найпростіший засіб автоматизованого відкриття форм, організації переходів між ними та іншими об'єктами бази даних — елемент керування «кнопка», добре відомий вам по роботі з ОС Windows та прикладними програмами. Кнопку можна розмістити на будь-якій формі так само, як і інші елементи керування. Під час розміщення буде запущено майстер створення кнопки, що дозволяє вказати дію, виконувану кнопкою, та настроїти її параметри.

Якщо створити порожню форму та розмістити на ній кнопки, які відкривають усі інші форми, отримаємо своєрідне кнопкове меню бази даних. Цей процес ми опишемо у вправі 7.2.

### Вправа 7.2

Створіть у базі даних **школа** форму-меню, на якій розміщуватимуться кнопки, що відкривають усі інші форми.

1. Виконайте команду **Создание формы в режиме конструктора** на вкладці **Формы** головного вікна бази даних в Access 2003 або клацніть кнопку **Конструктор форм** на стрічці **Создание** в Access 2007/2010. Буде відображено порожню форму **Форма1** у вікні конструктора.
2. Виберіть на панелі елементів або на стрічці **Конструктор** елемент керування **Кнопка** та клацніть те місце на формі, де кнопка розміщуватиметься. Буде запущено майстер створення кнопок.

3. У першому вікні майстра (рис. 7.4) вкажіть дію, яка виконуватиметься в результаті клацання кнопки. А саме, у категорії **Работа с формой** виберіть дію **Открыть форму**.

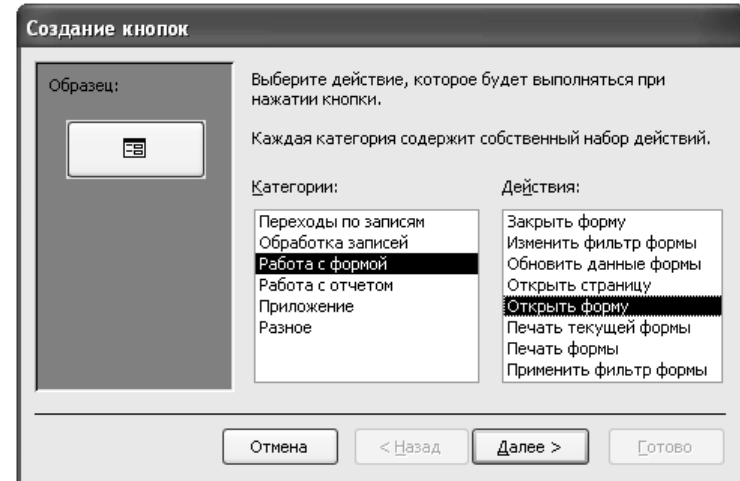


Рис. 7.4. Перше вікно майстра створення кнопок

4. У другому вікні майстра виберіть форму, яка відкриватиметься кнопкою. Нехай це буде форма **Учні**.
5. У третьому вікні встановіть перемикач **Открыть форму и показать все записи**.
6. У четвертому вікні (рис. 7.5) потрібно вказати, що відображатиметься на кнопці: рисунок (його файл слід вибрати за допомогою кнопки **Обзор**) або текст. Виберіть перемикач **Текст** і введіть текст **Учні**.
7. Клацніть кнопку **Готово** і вашу кнопку буде розміщено на формі.
8. Подібним чином на формі **Форма1** створіть кнопки для відкриття всіх інших форм бази даних **школа**. Кнопку для форми **Викладання** можна не створювати, оскільки ця форма є вкладеною до форми **Учителі**.



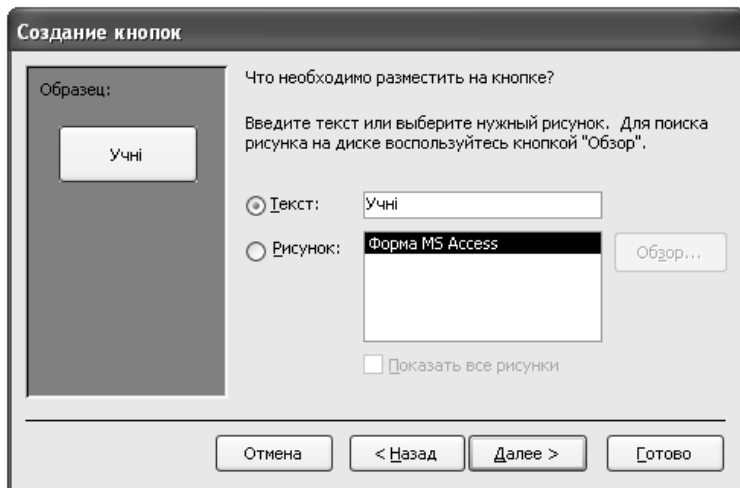





Рис. 7.5. Четверте вікно майстра створення кнопок

9. Зробіть розмір усіх кнопок однаковим. Для цього виділіть їх усі, клацніть якусь кнопку правою кнопкою миші і виберіть з контекстного меню команду **Размер** ▶ **по самому широкому**.
10. Вирівняйте усі кнопки зліва, виділивши їх та виконавши команду контекстного меню **Выровнять** ▶ **Слева** або **Выровнять** ▶ **по левому краю**.
11. Зробіть інтервал між кнопками однаковим: виділіть їх усі та клацніть кнопку  (Зробити інтервали по вертикалі рівними) на стрічці **Упорядочить** в Access 2007/2010 або виконайте команду **Формат** ▶ **Интервал по вертикали** ▶ **Сделать равным** в Access 2003.
12. Додайте над групою кнопок напис **Введення даних**, скориставшись кнопкою  (Напис).
13. Закрийте конструктор форми, надавши їй назву **Меню**, і перевірте дію створених вами кнопок.
14. Форма виглядатиме так, як зображено на рис. 7.6, а. Вона містить зайві елементи, непотрібні для форми-меню: інди-

катор поточного запису зліва та навігаційні елементи знизу. Щоб приховати їх, виконайте такі дії:

- а) перейдіть у вікно властивостей форми, клацнувши правою кнопкою миші позначку  у лівому верхньому куті форми та вибравши з контекстного меню команду **Свойства**;
- б) на вкладці **Макет** вікна властивостей для параметрів **Область выделения** та **Кнопки перехода** задайте значення **Нет**.

Готова форма має виглядати так, як на рис. 7.6, б.

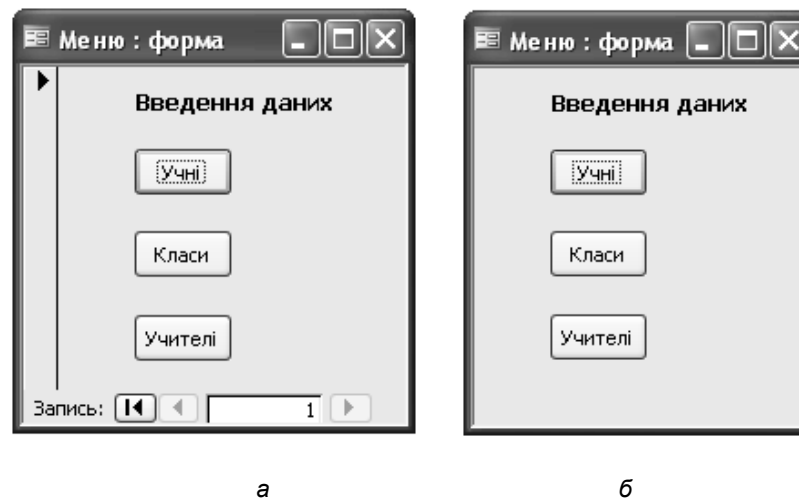


Рис. 7.6. Форма-меню: а — з зайвими елементами; б — у готовому вигляді

## Завдання 7.2

Додайте до кожної форми у базі даних **школа** кнопку **Вихід**, що закриває форму, та кнопку **Меню**, яка відкриває форму-меню.



### MS Access 2003


Було б зручно, якби форма-меню відкривалася автоматично після відкриття файлу бази даних. Щоб забезпечити таку поведінку, потрібно виконати команду **Сервіс** ► **Параметры запуска** і у вікні **Параметры запуска** вибрати потрібну форму зі списку **Вывод форми/страницы**. У цьому ж вікні можна скасувати відображення меню, панелей інструментів, вікна бази даних та інших елементів середовища СКБД так, що після відкриття вашої бази даних відобразатимуться лише її компоненти.

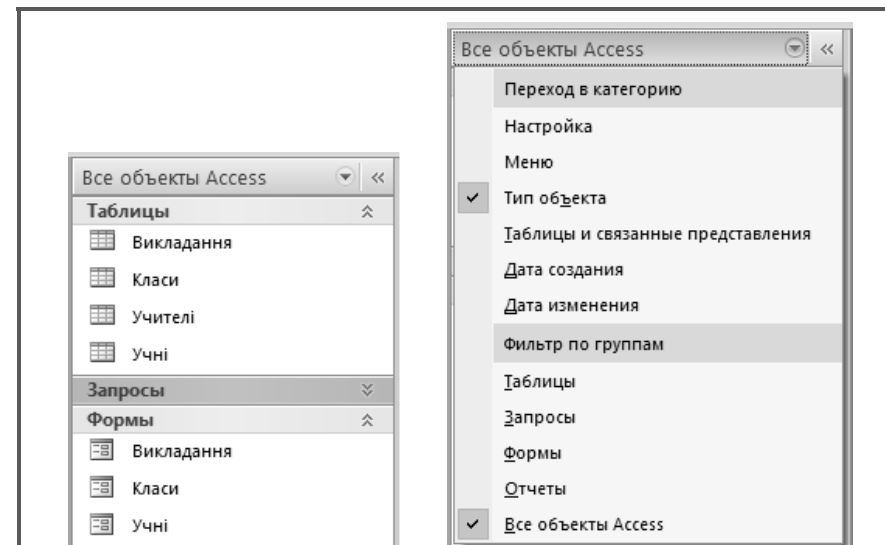
### Завдання 7.3 (тільки для MS Access 2003)

Забезпечте автоматичне відображення форми **Меню** після відкриття бази даних **школа**.

### MS Access 2007/2010

У MS Access 2007/2010 не передбачено засобів для автоматичного відкриття кнопочних меню, однак натомість є можливість керувати відображенням об'єктів в області переходів. У цій області об'єкти бази даних можуть відображатися різними способами, або, інакше кажучи, у різних *категоріях*. Назва поточної категорії відображається в заголовку області переходів. Об'єкти в кожній категорії можуть поділятися за групами, заголовки яких наведено на синіх смужках. Група може бути відкритою (тоді відображено всі її об'єкти) або закритою (тоді відображається лише заголовок групи). Щоб відкрити або закрити групу, слід клацнути на її заголовку кнопку  або  відповідно. Наприклад, на рис. 7.7, а зображено область переходів у категорії **Все объекты Access**, що містить три групи об'єктів: відкриті **Таблицы** та **Формы** і закриту **Запросы**.

Вибрати іншу категорію подання об'єктів, а також відобразити деякі чи всі групи в категорії можна за допомогою кнопки , розміщеної в заголовку області переходів справа. На рис. 7.7, б зображено меню, що відкривається цією кнопкою. У його верхній частині вибирають категорію, а в нижній — відображувану групу об'єктів.



а

б

Рис. 7.7. Область переходів: а — у категорії **Все объекты Access**; б — з контекстним меню заголовка

Користувач може створювати власні категорії, а в них — групи об'єктів. Для цього використовують вікно **Параметры переходов**, що відкривається однойменною командою контекстного меню заголовка області переходів. Зокрема можна створити категорію **Меню**, де відобразатимуться лише ті об'єкти, з якими має працювати користувач бази даних. Як це зробити, описано у вправі 7.3.

### Вправа 7.3 (тільки для MS Access 2007/2010)

Створіть в області переходів бази даних **школа** категорію **Меню** з однією групою **Формы**, що міститиме форми **Учні**, **Учителі** та **Класи**.

1. Клацніть правою кнопкою миші заголовок області переходів і виберіть з контекстного меню команду **Параметры**

переходов. Буде відображено вікно **Параметри переходов** (рис. 7.8).

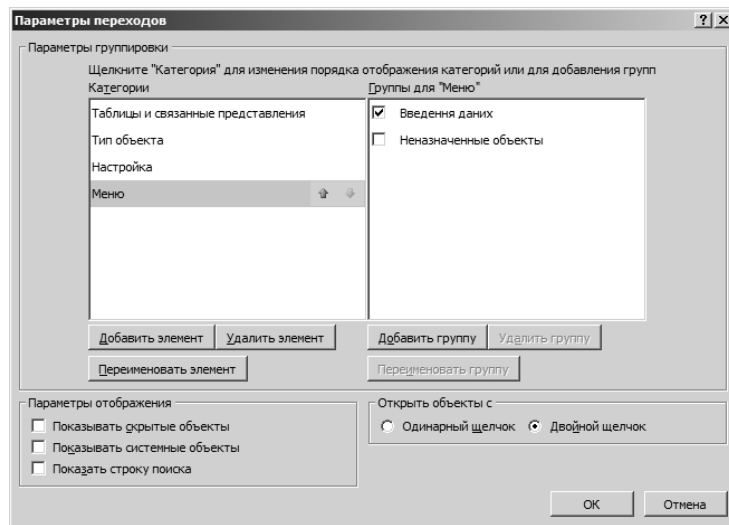


Рис. 7.8. Вікно Параметри переходов

- Щоб додати нову категорію, у лівій частині вікна клацніть кнопку **Добавить элемент** і введіть ім'я категорії — **Меню**.
- У категорії **Меню** створимо одну групу. Виділіть у лівій частині вікна цю категорію, а в правій клацніть кнопку **Добавить группу** й уведіть ім'я групи — **Введення даних**. Відобразатиметься також стандартна група **Неназначенные объекты** — вона містить об'єкти, які ви зможете додавати до власної групи.
- Клацніть кнопку **OK** і категорію буде створено. Відобразить її в області переходів за допомогою кнопки
- У створеній вами категорії ви побачите дві групи: порожню групу **Введення даних** та стандартну групу **Неназначенные объекты**, що містить всі об'єкти бази даних. Утримуючи клавішу **Ctrl**, виділіть значки форм **Учні**, **Учителі** та **Класи** у групі **Неназначенные объекты** і перетягніть їх на заголовок групи **Введення даних**.

- Приховайте групу **Неназначенные объекты**: клацніть її заголовок правою кнопкою миші й виберіть з контекстного меню команду **Скрыть**. Область переходів має набути такою вигляду, як на рис. 7.9.

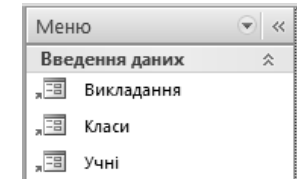


Рис. 7.9. Область переходів з меню

Якщо ви закриєте та знову відкриєте базу даних **школа**, область переходів міститиме створене вами меню форм.

## Завдання 7.4 (тільки для MS Access 2007/2010)

Додайте до категорії **Меню** групу **Викладання**, у якій зберіть всі форми і таблиці, що стосуються вчителів.

## Висновки

- Розкриті списки у формах створюють для введення значень зовнішніх ключів. Елементами розкритого списку є значення первинного ключа тієї таблиці, на яку посилається зовнішній ключ і, можливо, ще деяких її полів.
- Для швидкого доступу до потрібних користувачеві об'єктів бази даних і приховання інших об'єктів у MS Access 2003 на окремій формі створюють меню з кнопок, яке автоматично відображається після відкриття бази даних.
- У MS Access 2007/2010 для організації меню користувача створюють власні категорії і групи об'єктів в області переходів.

## Завдання для самостійного виконання

- У базі даних **школи** створіть форму для введення даних у таблицю **Директори**, а на ній — розкритий список для вибору директора серед учителів. Зробіть так, щоб у списку відображалася тільки прізвище та ім'я учителя.
- Для баз даних, створених у завданні для самостійного виконання з попереднього розділу, визначте, у які поля доцільно вводити дані за допомогою розкритих списків. Створіть два таких списки.

3. На будь-якій формі у базі даних **школа** створіть кнопку для дублювання записів та апробуйте її дію.

## Завдання для досліджень

1. Переставте ключове поле **паспорт** у таблиці **Учителі** з першого місця на якесь інше, після чого апробуйте дію списку, призначеного для вибору класного керівника. Він не працюватиме коректно. З'ясуйте причину помилки та виправте її. **ВКАЗІВКА.** Скористайтесь параметром **Присоединенный столбец** на вкладці **Данные** вікна властивостей списку.
2. У списках можуть відображатися не лише дані з таблиць, але і фіксовані набори значень, наприклад назви днів тижня. Навчіться створювати такі списки й використовувати їх для введення даних у таблиці.
- 3\*. У базі даних **школи** створіть форму для таблиці **Школи**, а в ній — розкритий список для вибору директора. У списку мають відображатися прізвища, імена та по батькові всіх директорів.

## Питання для роздумів

Нижче описано фрагменти предметних областей. Яким із них відповідають бази даних, де мають використовуватися списки? Для введення яких саме даних їх потрібно застосовувати?

- а) Зберігаються відомості про погодні явища і дні, коли вони спостерігалися.
- б) Є дані про бібліотечні фонди. Для кожної книжки відома її назва, шифр, кімната та шафа, де вона зберігається.
- в) У базі даних міститься інформація про людей. Потрібно зберігати відомості про те, хто є чіми батьками.
- г) Є відомості про лексичний склад певної мови. Щодо кожного слова потрібно зберігати відомості про те, до якої частини мови воно належить, які має суфікси, префікси, граматичні форми тощо.

## Розділ 8

# Вибирання даних

## Повторення

- ◆ Назвіть основні функції систем керування базами даних.
- ◆ Які операції з наборами однотипних об'єктів дає змогу виконувати табличний процесор?
- ◆ Для чого призначені і який формат мають функції для роботи з базою даних у середовищі табличного процесора?
- ◆ Як у табличному процесорі створюють розширений фільтр?

Згадаємо табличний процесор. Розглядаючи таблиці як набори однотипних об'єктів, ми застосовували до них такі операції, як фільтрування рядків, сортування, обчислення підсумкових характеристик для груп об'єктів тощо. Усі ці операції насправді є «рідними» для систем керування реляційними базами даних, звідки вони й були «запозичені» табличним процесором. Сьогодні ми розпочнемо знайомство з механізмами обробки даних, реалізованими в СКБД Microsoft Access.



## Вибирання даних з однієї таблиці

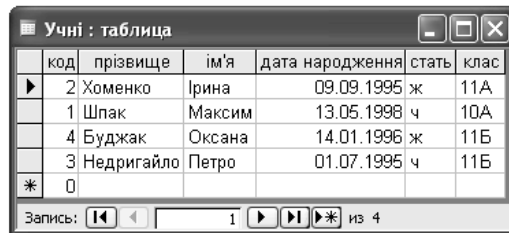
Над даними в одній таблиці реляційні СКБД дають змогу виконувати майже ті самі операції, що й табличний процесор. Насамперед це сортування, фільтрація, а також пошук і заміна даних. Можна також обчислювати підсумкові характеристики для груп записів, але в Microsoft Access цю функцію реалізують за допомогою засобів, які ми опишемо в наступних розділах. Зараз же розглянемо три операції, які виконують в режимі введення та редагування даних у таблиці.

## Сортування

*Сортуванням* називають розташування записів таблиці в порядку зростання чи спадання значень певного поля. Сортування, нагадаємо, найчастіше застосовують у випадках, коли:

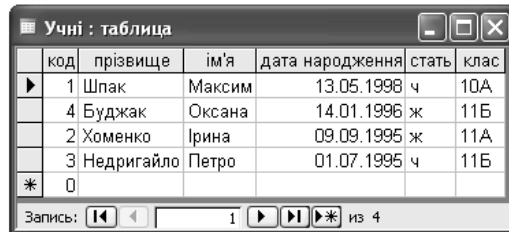
- ♦ необхідно дізнатися, які об'єкти мають малі, великі, найменші або найбільші значення тих чи інших параметрів (адже після сортування такі об'єкти розміщуватимуться на початку таблиці);
- ♦ потрібно згрупувати об'єкти за певним параметром, тобто розташувати поряд об'єкти з однаковими чи близькими його значеннями.

Щоб відсортувати записи таблиці за зростанням або спаданням значень якогось поля, потрібно встановити курсор у цьому полі та клацнути кнопку  (Сортування за зростанням) або  (Сортування за спаданням). На рис. 8.1, а зображено таблицю Учні, відсортовану за зростанням імен в алфавітному порядку, а на рис. 8.1, б — за спаданням дати народження.



код	прізвище	ім'я	дата народження	стать	клас
2	Хоменко	Ірина	09.09.1995	ж	11А
1	Шпак	Максим	13.05.1998	ч	10А
4	Буджак	Оксана	14.01.1996	ж	11Б
3	Недригайло	Петро	01.07.1995	ч	11Б
*	0				

а




код	прізвище	ім'я	дата народження	стать	клас
1	Шпак	Максим	13.05.1998	ч	10А
4	Буджак	Оксана	14.01.1996	ж	11Б
2	Хоменко	Ірина	09.09.1995	ж	11А
3	Недригайло	Петро	01.07.1995	ч	11Б
*	0				

б


Рис. 8.1. Сортування таблиці Учні: а — за зростанням імен, б — за спаданням дати народження

Для допитливих. Згадайте, як у табличному процесорі ми сортували таблицю за значеннями кількох полів: спочатку записи впорядковувалися в порядку зростання чи спадання значень одного поля, а потім кожна група записів з однаковим значенням цього поля сортувалася за значеннями іншого поля. Так само у Microsoft Access таблицю можна відсортувати за значеннями кількох суміжних полів, виділивши їх, а потім клацнувши кнопку сортування. Першим полем сортування буде те, яке розташоване лівіше.

## Фільтрація

Як і в табличному процесорі, *фільтрація* в СКБД Microsoft Access дозволяє відобразити тільки ті записи таблиці, що задовольняють певну умову. Щоб виконати фільтрацію, слід відкрити таблицю і клацнути кнопку  (Змінити фільтр), що розташована на панелі інструментів у MS Access 2003. У MS Access 2007/2010 аналогічну команду розміщено на стрічці Главная в області Сортировка и фильтр у меню кнопки Дополнительно.

У результаті буде відкрито вікно фільтра (рис. 8.2) з вкладками **Найти** та **Или**. Можливості цього засобу фільтрації такі самі, як і в розширеного фільтра в Microsoft Excel, але спосіб запису умови фільтрації дещо інший. Частина умови, що з'єднані сполучником «і», записують в одному рядку, а з'єднані сполучником «або» — на різних вкладках вікна фільтра. Наприклад, на рис. 8.2 зображено дві вкладки одного вікна фільтра, що реалізують, разом узяті, таку умову: *учні, що вчаться в 10А класі або вчаться в 11Б класі і народилися після 1.01.1996*.

Коли умову фільтрації введено, слід натиснути кнопку  **Применить фильтр**, і фільтр почне діяти. Так, на рис. 8.3 показано результат застосування до таблиці Учні умов фільтрації, зображених на рис. 8.2. Щоб побачити таблицю у початковому вигляді, цю кнопку потрібно відтиснути.

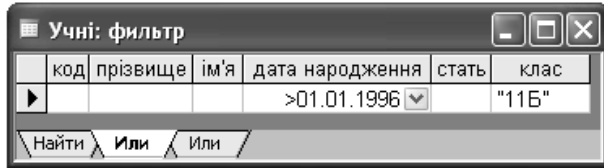
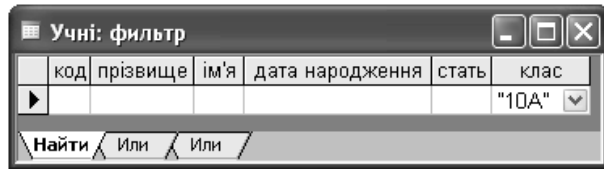


Рис. 8.2. Вікно фільтра

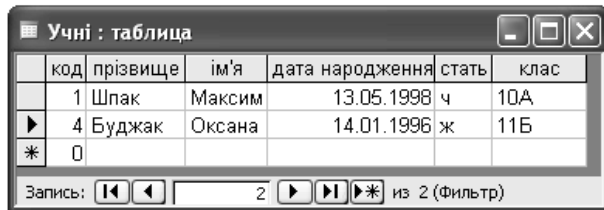


Рис. 8.3. Результат фільтрації

В умовах фільтрації можна використовувати символи підстановки. Наприклад, якщо ввести в поле клас вікна фільтра таблиці *Учні* текст 11\*, то буде відображено відомості про учнів усіх одинадцятих класів — як 11А, так і 11Б.

Щоб очистити умову фільтрації, потрібно клацнути вікно фільтра правою кнопкою миші і в меню, що з'явиться, вибрати команду **Очистити бланк**. За допомогою команди **Удалить вкладку** можна видалити лише поточну вкладку вікна фільтра, тобто частину умови, з'єднану з іншими частинами сполучником «або».

Найпростіший тип фільтра — це *фільтр за виділенням*. Під час його застосування значення в клітинці, де встановлено курсор, вважається значенням-зразком, а поле, де встановлено курсор, — полем фільтрації. У результаті фільтрації відобража-

ються ті записи, які містять у полі фільтрації таке саме значення-зразок або подібне до нього.

Цей фільтр у MS Access 2003 і MS Access 2007/2010 застосовують дещо по-різному.

### MS Access 2003

Для того, щоб застосувати фільтр за виділенням, досить клацнути кнопку . Буде відображено всі записи, що містять в полі фільтрації значення-зразок. Можна взяти за зразок значення з кількох полів, виділивши суміжні клітинки таблиці за допомогою табличного курсору .

### MS Access 2007/2010

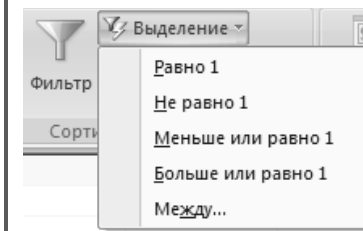


Рис. 8.4. Меню кнопки **Выделение**

Припустимо, у певній клітинці міститься значення-зразок, а вам потрібно відобразити записи, які в тому ж полі містять таке саме значення або значення, більші чи менші за нього, такі, що включають значення-зразок тощо. Тоді слід установити курсор на значення-зразок і з меню кнопки **Выделение** вибрати один зі способів порівняння шуканого значення зі значенням-зразком (рис. 8.4).

## Пошук і заміна

Засіб автоматизованого пошуку та заміни даних у Microsoft Access дуже подібний до аналогічного засобу табличного процесора Microsoft Excel. Його відкривають натисканням клавіш **Ctrl+F** (пошук) чи **Ctrl+H** (заміна). Використовуючи елементи керування зображеного на рис. 8.5 вікна **Поиск и замена**, ви можете шукати значення всього поля або його частини, з урахуванням регістра літер або без, в окремому полі або в усій таблиці тощо.

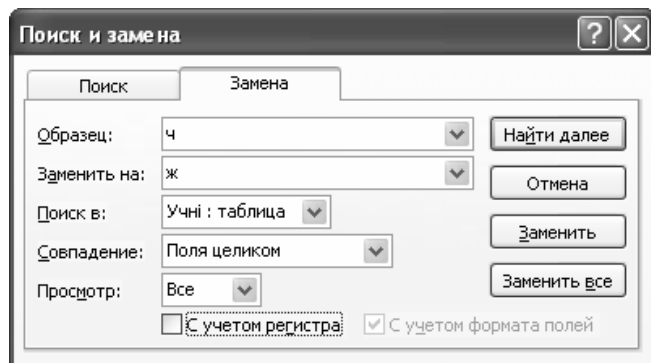


Рис. 8.5. Вікно Поиск и замена

## Завдання 8.1

За допомогою фільтрації відобразіть відомості про всіх учнів 10А класу і тих учнів 11Б класу, які народилися після 1.01.1996. Спочатку сформулюйте умову фільтрації, якій має відповідати окремий учень.

## Вибір даних з кількох таблиць

Клієнти баз даних — люди або програми — часто потребують інформації, яку неможливо знайти в якійсь одній таблиці. Припустимо, потрібно дізнатися прізвища та імена вчителів, які викладають в 11А класі. Очевидно, що цю інформацію не вдасться відобразити за допомогою фільтра, застосованого до таблиці вчителів, адже в ній немає жодної інформації про класи. З іншого боку, у таблицях Класи та Викладання немає інформації про прізвища та імена вчителів, тому фільтрація цих таблиць також не допоможе. У подібних випадках стануть у нагоді *запити* — універсальний засіб виконання майже будь-яких завдань з обробки даних у БД. Словом «запит» називають також саму задачу на кшталт «визначити, які вчителі викладають в 11А класі»; результатом її розв’язання буде список учителів, а також текст розв’язання, записаний мовою SQL або поданий у спеціальній екранній формі.

## Схема роботи з запитом

Роботу з будь-яким запитом можна поділити на три основні етапи:

- ◆ спочатку запит *формулюють*, тобто описують словесно;
- ◆ далі запит *створюють*, або *реалізують* у СКБД — «перекладають» текст запиту мовою SQL або вводять його в екранну форму;
- ◆ нарешті запит *виконують*, тобто дають СКБД команду отримати розв’язок задачі з обробки даних.

Запит створюють один раз, а виконують багаторазово. Кожен раз результати виконання можуть бути різними, залежно від того, які дані введено в таблиці, що ними оперує запит.

Зазначимо, що результатом виконання запиту в реляційній БД також є таблиця. Однак це не одна з тих таблиць, які входять до складу БД. Це так звана *віртуальна таблиця*, що існує не тривалий час, а саме стільки, скільки потрібно клієнту, але не довше, ніж протягом одного сеансу роботи з СКБД. Ця таблиця зберігається лише в оперативній пам’яті, і коли ви завершуєте роботу з СКБД, усі віртуальні таблиці видаляються, проте зберігаються SQL-тексти запитів і для отримання віртуальної таблиці під час наступного сеансу роботи з СКБД достатньо просто виконати запит.

**Для допитливих.** Той факт, що результат виконання запиту є таблицею, — вкрай важливий, адже це означає, що одні запити можна використовувати як параметри інших, тобто будувати запити на основі запитів, а не таблиць. Інакше кажучи, як віртуальні таблиці, так і ті, що зберігаються в БД постійно, можна вважати однаковиими з точки зору конструювання запитів. У теорії реляційних баз даних така властивість мов маніпулювання даними називається *реляційною замкненістю*.

## Створення запитів у СКБД Microsoft Access

У Microsoft Access передбачено два засоби автоматизованого створення запитів: майстер і конструктор. Можливості першого

із них вкрай обмежені, тому ми розглядатимемо тільки другий і зробимо це на прикладі запиту *визначити прізвища та імена вчителів, які викладають в 11А класі*.

Отже, для створення запиту у MS Access 2003 необхідно у головному вікні бази даних вибрати в меню об'єктів пункт **Запросы** і двічі клацнути посилання **Создание запроса в режиме конструктора**, а в MS Access 2007/2010 — клацнути кнопку **Конструктор запросов** в області **Другие** на вкладці **Создание**. Буде відображено вікно *конструктора запиту*, що називається **Запрос 1: запрос на выборку**, а також вікно **Добавление таблицы** (рис. 8.6).

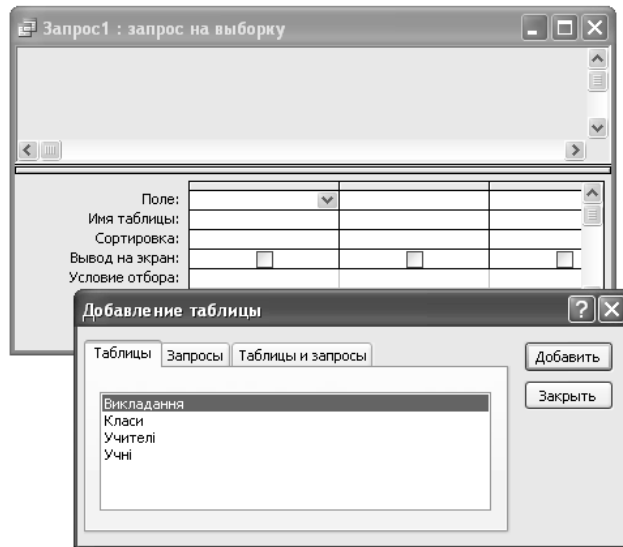


Рис. 8.6. Створення запиту

У вікні **Добавление таблицы** потрібно, утримуючи клавішу **Ctrl**, вибрати назви таблиць, дані з яких використовуватимуться в запиті, клацнути кнопку **Добавить**, а потім — **Закерть**. Для нашого запиту достатньо двох таблиць, **Учителі** та **Викладання**, оскільки назви класів містяться в таблиці **Викладання**, а прізвища та імена вчителів, які в цих класах викладають, — у таблиці **Учителі**. Якщо додати таблицю **Класи**, це не буде помилкою, але призведе до зайвого ускладнення запиту, оскільки

класи зв'язані з учителями через таблицю **Викладання**, яку доведеться додавати все одно.

У верхній частині вікна конструктора запиту відображається фрагмент схеми БД, що містить ті таблиці, які використовуються в запиті (рис. 8.7). У нижній частині вікна розміщено *бланк запиту* — прототип таблиці, яка відобразиться в результаті виконання запиту (рис. 8.7). У стовпцях цієї таблиці розміщують поля, перетягуючи їх з верхньої частини, а також задають параметри відображення полів та умови відбору їхніх значень.

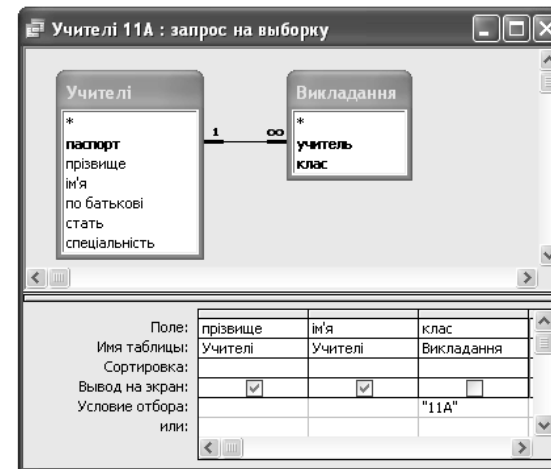


Рис. 8.7. Вікно конструктора запиту «Визначити прізвища та імена вчителів, які викладають в 11А класі»


Призначення рядків бланка запиту таке:


- ◆ **Поле** — назва поля, що відобразиться в таблиці-результаті запиту або значення якого мають задовольняти умову;
- ◆ **Имя таблицы** — назва таблиці, у якій міститься поле;
- ◆ **Сортировка** — визначення способу сортування рядків у таблиці-результаті;
- ◆ **Вывод на экран** — прапорець, який слід встановити, якщо поле відобразиться в таблиці-результаті;
- ◆ **Условие отбора** — умова, яку мають задовольняти значення полів;



- ♦ **или** — частина умови, з'єднана з іншими частинами сполучником «або».

Бланк нашого запиту має виглядати так, як на рис. 8.7: відображаються поля прізвище та ім'я таблиці Учителі, а значенням поля клас таблиці Викладання має бути 11А.

Щоб виконати цей запит, потрібно клацнути кнопку  (Запуск) на панелі інструментів. Яким має бути результат, показано на рис. 8.8.

Після того як ви переглянете результати запиту, вікно таблиці результатів потрібно закрити кнопкою . Буде відображено запитання, чи зберігати зміни макету або структури об'єкта **Запрос 1**. Клацніть кнопку **Да** і у вікні, що з'явиться, введіть назву запиту, наприклад Учителі 11А. Коли ви клацнете кнопку ОК, запит відобразиться в розділі запитів у головному вікні бази даних (рис. 8.9) або в категорії **Все объекты Access** в області переходів.

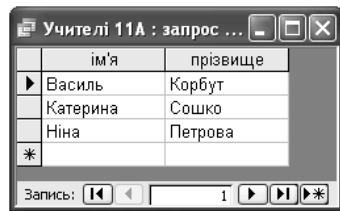


Рис. 8.8. Результат виконання запиту

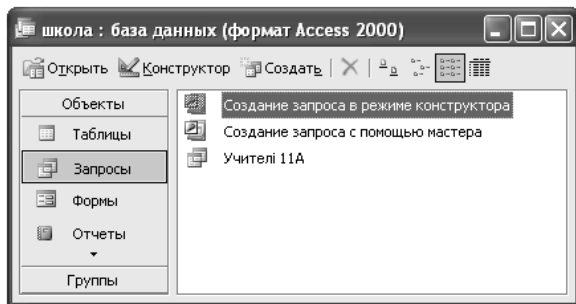




Рис. 8.9. Відображення запиту в головному вікні бази даних

## Редагування запитів

Якщо в головному вікні бази даних Access 2003 вибрати в меню об'єктів команду **Запросы**, у правій частині цього вікна відобразяться всі наявні в базі запити. У MS Access 2007/2010 усі запити можна побачити в групі **Запросы** категорії **Все объекты Access** в області переходів. За потреби змінити якийсь запит клацніть його правою кнопкою миші та виберіть з контекстного меню команду **Конструктор**. Відкриється знайоме вам вікно конструктора запиту, у якому можна змінити як набір таблиць, що використовуються в запиті, так і його бланк.

- ♦ Щоб видалити з запиту таблицю, клацніть її у верхній частині вікна конструктора правою кнопкою та виберіть у контекстному меню команду **Удалить таблицу**.
- ♦ Щоб додати певну таблицю, скористайтеся кнопкою  (Відобразити таблицю).
- ♦ Для видалення з запиту зв'язка між таблицями його потрібно клацнути правою кнопкою миші та вибрати з меню, що з'явиться, команду **Удалить**.
- ♦ Якщо знадобиться видалити з бланка запиту поле, клацніть його заголовок (тонку сіру смужку над назвою поля) правою кнопкою миші та виберіть з меню, що відкриється, команду **Вырезать**.
- ♦ Щоб змінити параметри полів у бланку, введіть потрібні значення з клавіатури або виберіть їх зі списку, що розкривається кнопкою  у клітинках рядків **Поле**, **Имя таблицы**, **Сортировка**.

## Завдання 8.2

Створіть та виконайте запит «Визначити прізвища та імена вчителів, які викладають в 11А класі».

## Завдання 8.3

Створіть та виконайте запит «Визначити прізвище та спеціальність класного керівника 11Б класу».

## Приклади запитів

### Приклад 1

Розглянемо трохи складніший запит: *визначити імена та прізвища учнів, яким викладає Сошко Катерина Миколаївна*. У цьому запиті потрібно використати всі чотири таблиці, які є в базі даних **школа**: з таблиці **Учні** взяти прізвища та імена, на значення полів таблиці **Учителі** накласти умову прізвище = Сошко, ім'я = Катерина, по батькові = Миколаївна, а скориставшись таблицями **Викладання** та **Класи**, з'єднати вчителя з учнями.

Який вигляд матиме вікно конструктора цього запиту, показано на рис. 8.10. Зауважте, що зв'язок «один-до-одного» між таблицями **Класи** та **Учителі** було видалено, оскільки його зміст — «вчитель є класним керівником» — ніяк не відображається в умові запиту. Якби ми цей зв'язок залишили, то реалізували б інший запит, а саме «Визначити імена та прізвища учнів, у яких викладає та є класним керівником Сошко Катерина Миколаївна». Нагадаємо, що зв'язок з бланка запити видаляють так само, як і зі схеми БД: на зв'язку потрібно клацнути правою кнопкою миші та вибрати у меню, що з'явиться, команду **Удалить**.

Якщо ви правильно реалізували цей запит і ввели в базу дані, вказані у завданнях до розділів 4 і 5, то в результаті його виконання маєте отримати Максим Шпак та Хоменко Ірина.

### Приклад 2

*Визначити, у яких класах навчаються хлопці, що народилися у 1996 році.*

Для створення цього запиту достатньо однієї таблиці — **Учні**, у якій міститься інформація і про класи. Тому цей запит загалом можна реалізувати за допомогою фільтра. Щоправда, нам потрібно відобразити значення одного поля — класи, а після фільтрації відобразатимуться всі поля. Вікно конструктора цього запиту наведено на рис. 8.11.

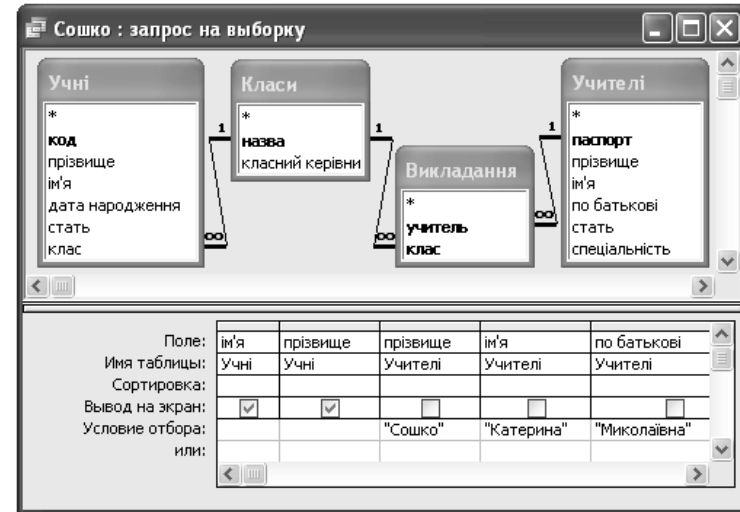


Рис. 8.10. Вікно конструктора запиту «визначити імена та прізвища учнів, у яких викладає Сошко Катерина Миколаївна»

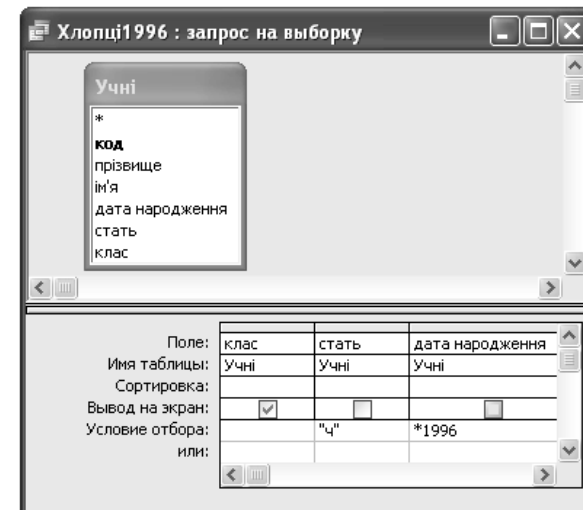


Рис. 8.11. Вікно конструктора запиту «Визначити, у яких класах навчаються хлопці, що народилися у 1996 році»

Зверніть увагу на такі обставини:

- ◆ На записи таблиці *Учні* накладено дві умови, з'єднані сполучником «і»: *стать* має бути чоловічою і *дата народження* — 1996 рік. Тому ці умови розміщено в одному рядку **Условие отбора**.
- ◆ Умову «народилися у 1996 році» записано як \*1996. Буквально цей запис можна інтерпретувати так: значення поля *дата народження* закінчується числом 1996. Символ «\*» тут є знаком підстановки, що позначає довільний набір символів.

### Приклад 3

Відобразити всю інформацію про класних керівників 10А та 11Б класів.

Для реалізації цього запиту знадобляться таблиці *Класи* та *Учителі*, а таблиця *Викладання* буде зайвою, оскільки про викладання в умові запиту не йдеться. Вікно конструктора запиту зображено на рис. 8.12.

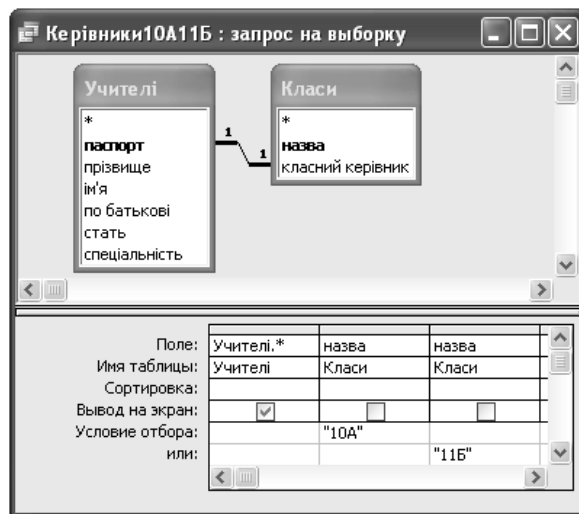


Рис. 8.12. Вікно конструктора запиту «Відобразити всю інформацію про класних керівників 10А та 11Б класів»

І в цьому запиті варто звернути увагу на дві обставини: «Всю інформацію» означає «значення всіх полів», але на бланку запиту всі поля таблиці *Учителі* перелічувати не потрібно. Замість цього достатньо записати в імені поля символ \*, який і означає «всі поля».

Умову запиту можна переформулювати так: «Відобразити всю інформацію про вчителів, які є класними керівниками 10А або 11Б класу». Тобто, хоча в умові вжито сполучник *та*, насправді її частини з'єднані сполучником *або* і тому записані в різних рядках: **Условие отбора** *та* **или**.

### Завдання 8.4

Створіть та виконайте запит «Отримати всю інформацію про вчителів, які викладають учням, що народилися після 1 січня 1996 року».

Для допитливих. Існують засоби автоматизованого створення запитів, значно потужніші, ніж конструктор у СКБД Microsoft Access. Так, у СКБД Paradox реалізовано графічну мову маніпулювання даними, що називається QBE (Query by Example — запит за зразком). Її можливості навіть ширші за можливості мови SQL.

### Висновки

- ◆ Під сортуванням розуміють розташування записів таблиці у порядку зростання чи спадання значень якогось поля.
- ◆ Фільтрація дає змогу відобразити тільки ті записи таблиці, що задовольняють певну умову.
- ◆ Запити — універсальний засіб виконання майже будь-яких завдань з обробки даних у БД.
- ◆ Запит спочатку формулюють, тобто описують словесно, потім створюють, або реалізують, у СКБД — «перекладають» текст запиту мовою SQL або вводять його в екранну форму, і, нарешті виконують, тобто дають СКБД команду отримати розв'язок задачі з обробки даних.

- ◆ Результатом виконання запиту в реляційній БД є віртуальна таблиця, яка існує стільки, скільки потрібно клієнту.
- ◆ Створюючи запит, у вікні конструктора необхідно вказати таблиці, дані з яких використовуватимуться, поля, значення яких потрібно відобразити на екрані, та умови, яким мають відповідати значення цих або інших полів.

## Завдання для самостійного виконання

1. За допомогою фільтрації відобразіть відомості лише про тих учителів, які мають паспорт серії СО та викладають фізику або математику.
2. Реалізуйте та виконайте такі запити:
  - Відобразити всі дані про учнів, у яких класний керівник — Корбут Василь Петрович.
  - Визначити спеціальності вчителів, які викладають в 11 класах.
3. Для кожної з баз даних, створених у завданнях для самостійного виконання з попередніх розділів, сформулюйте, реалізуйте та виконайте по одному запиту, у якому використовується більше однієї таблиці.
- 4\*. Реалізуйте та виконайте такі запити:
  - Визначити прізвища та дати народження учнів, яким викладає Корбут Василь Петрович або Сошко Катерина Миколаївна.
  - Визначити прізвища та дати народження учнів, яким викладають Корбут Василь Петрович та Сошко Катерина Миколаївна.
  - Визначити прізвища та дати народження учнів, яким не викладає Корбут Василь Петрович.

## Питання для роздумів

1. Чому можливості вибирання даних у СКБД потужніші, ніж у табличному процесорі?

- 2\*. Чим відрізняється запит «Визначити класи, у яких не викладає Сошко Катерина Миколаївна» від запиту «Визначити класи, у яких викладає не Сошко Катерина Миколаївна»? Наведіть приклади даних, на яких ці запити даватимуть різні результати.

## Завдання для досліджень

1. Навчіться використовувати майстер побудови запитів у Microsoft Access та визначте, які різновиди запитів він дозволяє створювати. Наведіть приклад запиту, який неможливо створити за допомогою майстра, але можна реалізувати в конструкторі запитів.
- 2\*. Сформулюйте для бази даних **школа** запит, який неможливо реалізувати засобами конструктора запитів.

## Розділ 9°

# Основи мови запитів

### Повторення

- ◆ Що таке мова маніпулювання даними?
- ◆ Яка мова маніпулювання даними найбільш поширена в реляційній моделі?
- ◆ Який вигляд має вікно конструктора запитів?
- ◆ Що таке віртуальна таблиця?

Як ми вже зазначали, практично в усіх реляційних СКБД для маніпулювання даними застосовують мову SQL (англ. Structured Query Language — мова структурованих запитів), а такі автоматизовані засоби створення запитів, як ми розглядали в попередньому розділі, представлені лише в поодиноких системах керування базами даних. Тобто основний спосіб конструювання запитів до баз даних — це їх запис у текстовому вигляді, чимось подібний до написання невеличких програм. Однак мова SQL не є мовою програмування, вона декларативна, тобто дозволяє користувачу описати, **що** він хоче отримати, не описуючи, **як** саме комп'ютер має обчислити потрібний результат. Тому писати запити мовою SQL значно легше, ніж програми будь-якою мовою програмування. Загалом є кілька різновидів запитів: на вибирання даних, їх додавання, видалення, оновлення та деякі інші. Сьогодні ви навчитеся описувати мовою SQL нескладні запити на вибирання даних.

### Загальна структура SQL-запиту

Ви вже вмієте створювати *запити на вибирання даних*, або *вибіркові запити* за допомогою конструктора запитів MS Access.

Відмінною рисою цього типу запитів є те, що вони не змінюють дані в базі, а лише вибирають їх з таблиць за певними умовами. Результатом виконання будь-якого вибіркового запиту, нагадаємо, є віртуальна таблиця, що існує нетривалий час, поки ви її не закриєте.

Зрозуміти структуру вибіркового SQL-запиту найлегше на конкретному прикладі, розглянутому у вправі 9.1.

### Вправа 9.1

Запишіть мовою SQL і виконайте запит *визначити прізвища та імена учнів-хлопців*.

1. Відкрийте конструктор запитів. У вікні **Добавление таблицы** виберіть таблицю **Учні**, оскільки запит стосується учнів, після чого клацніть кнопку **Добавить** і закрийте це вікно.
2. У Access 2003 клацніть правою кнопкою миші будь-яке місце вікна конструктора, крім бланка запиту, й виберіть з контекстного меню команду **Режим SQL**, а в Access 2007/2010 з меню кнопки **Режим** на стрічці **Конструктор** виберіть команду **Режим SQL**. Буде відображено заготовку SQL-коду запиту:

```
SELECT  
FROM Учні;
```

У цій заготовці ви бачите два ключових слова: SELECT і FROM. Після слова FROM вказують назви таблиць, з яких вибиратимуться дані. У нашому випадку це буде таблиця **Учні**; її назву вже введено автоматично.


3. Після слова SELECT через кому запишіть назви полів, значення яких відобразатимуться. У даному запиті це поля **прізвище** та **ім'я**. Оскільки назва поля **ім'я** містить символ апострофа, її потрібно взяти у квадратні дужки:

```
SELECT прізвище, [ім'я]  
FROM Учні;
```

Зазначимо, що у квадратні дужки беруться також назви, які містять пробіл.

4. В кінці запиту (але до символу `;`) запишіть ключове слово `WHERE`, а після нього — умову відбору записів (серед усіх учнів нам потрібно відібрати хлопців). Остаточний вигляд запиту буде таким:

```
SELECT прізвище, [ім'я]
FROM Учні
WHERE стать="ч";
```

5. Запустіть запит на виконання кнопкою  (Виконати) та переконайтеся, що в таблиці результатів є відомості тільки про хлопців.
6. Закрийте вікно запиту, зберігши його під іменем учні-хлопці.

Загалом вибірковий SQL-запит має такий формат:

```
SELECT список полів, значення яких потрібно отримати
FROM список таблиць, з яких вибираються дані
WHERE умова, яку мають задовольняти записи,
      що вибираються
```

Зазначимо, що вирази, які починаються з ключових слів, часто називають *фразами*, наприклад *фраза* `SELECT`. Регістр ключових слів неважливий, але їх прийнято записувати великими літерами.

Зауважимо також, що мова SQL *англійована*, тобто її вирази дещо нагадують англійські речення. Слово `SELECT` перекладається як «вибрати», слово `FROM` — «з», а `WHERE` — «де», або «для яких». У цілому простий SQL-запит потрібно читати за такою схемою.

Запит `SELECT x FROM y WHERE z` слід інтерпретувати так: «вибрати поля *x* тих записів таблиці *y*, які задовольняють умову *z*».

**Для допитливих.** Якщо запит із вправи 9.1 скласти в конструкторі, а потім перейти в режим SQL, то ми побачимо такий текст, автоматично сконструйований СКБД:

```
SELECT Учні.прізвище, Учні.[ім'я]
FROM Учні
WHERE ((Учні.стать)="ч");
```

За змістом він нічим не відрізняється від тексту, записаного нами в SQL-редакторі, але виглядає складнішим через те, що під час автоматичної побудови SQL-виразів MS Access вводить деякі зайві позначення. Так, після слова `WHERE` поставлено непотрібні дужки, а перед назвами атрибутів записано назву таблиці з символом крапки. Вираз `Учні.прізвище` можна читати як «поле прізвище таблиці `Учні`». Проте якщо у запиті всього одна таблиця, то поле прізвище ніякій іншій таблиці не може належати і тому специфікатор `Учні.` можна опустити.

## Завдання 9.1

Запишіть мовою SQL запит *визначити прізвища та номери паспортів учителів-математиків*. Як перейти в режим введення SQL-коду, описано у вправі 9.1.

## З'єднання таблиць

Розглянемо запит *визначити прізвища та імена вчителів, що викладають в 11А класі*, який ми вже реалізовували за допомогою конструктора запитів (див. рис. 8.7). Якщо створити цей запит у конструкторі, а потім відобразити його SQL-код та видалити зайві дужки і специфікатори, результат буде таким:

```
SELECT прізвище, [ім'я]
FROM Учителі INNER JOIN Викладання ON
      Учителі.паспорт = Викладання.учитель
WHERE клас="11А"
```

У фразі FROM ми бачимо не назву таблиці і не список назв, а оператор INNER JOIN (у перекладі з англійської — внутрішнє з'єднання), призначений для з'єднання таблиць. Загальний синтаксис цього оператора такий:

Таблиця1 INNER JOIN Таблиця2 ON умова

Умова зазвичай стосується обох таблиць, вказаних до і після слів INNER JOIN (у прикладі вище — таблиць Учителі та Викладання). Оператор «зчіплює» ті записи таблиці 1 і таблиці 2, які, разом узяті, відповідають умові. У результаті з таких зчіпок формується нова таблиця, до якої вже застосовується відбір рядків згідно з фразою WHERE, а потім — відбір стовпців згідно з фразою SELECT.

Опишемо детальніше алгоритм, за яким СКБД виконує запит.

1. Виконується оператор INNER JOIN у фразі FROM. А саме, зчіплюються всі такі пари записів з таблиць Учителі та Викладання, для яких виконується рівність `Учителі.паспорт = Викладання.учитель`, тобто значення поля `паспорт` у таблиці `Учителі` збігається зі значенням поля `учитель` у таблиці `Викладання` (рис. 9.1, а). У результаті отримуємо таблицю, що містить усі такі зчіпки (рис. 9.1, б).

Оскільки у зв'язку між таблицями `Учителі` та `Викладання` таблиця `Учителі` є головною, а `Викладання` — підлеглою, то оператор `Учителі INNER JOIN Викладання` можна інтерпретувати так: «до кожного вчителя дописати інформацію про його викладання».

2. З таблиці, зображеної на рис. 9.1, б відбираються записи за умовою `клас="11А"`, тобто ті записи, які в полі `клас` містять значення `11А`. У результаті отримуємо таблицю, зображену на рис. 9.2.
3. У таблиці, зображеній на рис. 9.2, залишаються тільки два поля, вказані після слова `SELECT`: `прізвище` та `ім'я`. Отримуємо таблицю, зображену на рис. 9.3, — це і є остаточний результат виконання запиту.

клас	учитель
11Б	СН 410268
11А	СО 211517
11Б	СО 211517
10А	СР 652320
11А	СР 652320
10Б	КН 200125
11Б	КН 200125
10А	СО 927453
11А	СО 927453
11Б	СО 927453

паспорт	прізвище	ім'я	по батькові	стать	спеціальність
СН 410268	Михайлюк	Дмитро	Семенович	ч	математик
СО 211517	Корбут	Василь	Петрович	ч	математик
СР 652320	Сошко	Катерина	Миколаївна	ж	біолог
КН 200125	Томчишин	Віктор	Георгійович	ч	історик
СО 927453	Петрова	Ніна	Володимирівна	ж	фізик

а

паспорт	прізвище	ім'я	по батькові	стать	спеціальність	учитель	клас
СН 410268	Михайлюк	Дмитро	Семенович	ч	математик	СН 410268	11Б
СО 211517	Корбут	Василь	Петрович	ч	математик	СО 211517	11А
СО 211517	Корбут	Василь	Петрович	ч	математик	СО 211517	11Б
СР 652320	Сошко	Катерина	Миколаївна	ж	біолог	СР 652320	10А
СР 652320	Сошко	Катерина	Миколаївна	ж	біолог	СР 652320	11А
КН 200125	Томчишин	Віктор	Георгійович	ч	історик	КН 200125	10Б
КН 200125	Томчишин	Віктор	Георгійович	ч	історик	КН 200125	11Б
СО 927453	Петрова	Ніна	Володимирівна	ж	фізик	СО 927453	10А
СО 927453	Петрова	Ніна	Володимирівна	ж	фізик	СО 927453	11А
СО 927453	Петрова	Ніна	Володимирівна	ж	фізик	СО 927453	11Б

б

Рис. 9.1. З'єднання таблиць `Учителі` та `Викладання`: а — зчеплення записів; б — результат з'єднання

паспорт	прізвище	ім'я	по батькові	стать	спеціальність	учитель	клас
СО 211517	Корбут	Василь	Петрович	ч	математик	СО 211517	11А
СР 652320	Сошко	Катерина	Миколаївна	ж	біолог	СР 652320	11А
СО 927453	Петрова	Ніна	Володимирівна	ж	фізик	СО 927453	11А

Рис. 9.2. Відбір записів зі з'єднаної таблиці

прізвище	ім'я
Корбут	Василь
Сошко	Катерина
Петрова	Ніна

Рис. 9.3. Проекція таблиці на поля, вказані після слова SELECT

У загальному випадку SQL-запит виконується так:

1. Виконується операція з'єднання таблиць із фрази FROM.
2. До отриманої таблиці застосовується операція вибірки: відбираються ті записи, що задовольняють умову, вказану у фразі WHERE.
3. Таблиця, отримана на попередньому кроці, проектується на поля, вказані у фразі SELECT, тобто ці поля залишаються, а всі інші відкидаються.

Більшість вибірових запитів до реляційних баз даних (але не всі) виконуються саме за цією схемою: з'єднання таблиць — вибірка записів — проекція на поля.

Для допитливих. З'єднання, вибірка і проекція — три з восьми операцій реляційної алгебри Кодда. Як уже згадувалося в розділі 1, реляційна алгебра — це мова маніпулювання даними, запропонована автором реляційної моделі Е. Коддом. Багато її рис увібрала в себе мова SQL.

## Завдання 9.2

Запишіть оператори INNER JOIN, які дозволять отримати в базі даних школа такі з'єднання таблиць, як показано на рис. 9.4, а–б.

назва	класний керівн	паспорт	прізвище	ім'я	по батькові	стать	спеціальність
10А	СН 410268	СН 410268	Михайлюк	Дмитро	Семенович	ч	математик
11Б	СО 211517	СО 211517	Корбут	Василь	Петрович	ч	математик
10Б	КН 200125	КН 200125	Томчишин	Віктор	Георгійович	ч	історик
11А	СО 927453	СО 927453	Петрова	Ніна	Володимирівна	ж	фізик

а

код	прізвище	ім'я	дата народжж	стать	Учні.клас	назва	класний керівн	учитель	Висла
1	Шлак	Максим	13.05.1998	ч	10А	10А	СР 652320	СР 652320	10А
1	Шлак	Максим	13.05.1998	ч	10А	10А	СР 652320	СО 927453	10А
2	Хоменко	Ірина	09.09.1995	ж	11А	11А	СО 927453	СР 652320	11А
2	Хоменко	Ірина	09.09.1995	ж	11А	11А	СО 927453	СО 211517	11А
2	Хоменко	Ірина	09.09.1995	ж	11А	11А	СО 927453	КН 200125	11А
3	Недригайло	Петро	01.07.1995	ч	11Б	11Б	СО 211517	СН 410268	11Б
4	Бужжак	Оксана	14.01.1996	ж	11Б	11Б	СО 211517	СО 211517	11Б
3	Недригайло	Петро	01.07.1995	ч	11Б	11Б	СО 211517	СО 211517	11Б
4	Бужжак	Оксана	14.01.1996	ж	11Б	11Б	СО 211517	СО 211517	11Б
3	Недригайло	Петро	01.07.1995	ч	11Б	11Б	СО 211517	СО 927453	11Б
4	Бужжак	Оксана	14.01.1996	ж	11Б	11Б	СО 211517	СО 927453	11Б

б

Рис. 9.4. З'єднання таблиць

## Завдання 9.3

Реалізуйте мовою SQL запит визначити, у якій класі класним керівником є Сошко Катерина Миколаївна.

Для допитливих. З'єднувати таблиці можна і без оператора INNER JOIN. Для цього слід перелічити їхні назви через кому у фразі FROM, а у фразі WHERE записати умову з'єднання записів, долучивши її до інших умов сполучником AND. Наприклад, запит визначити прізвища та імена вчителів, що викладають в 11А класі можна реалізувати так:

```
SELECT прізвище, [ім'я]
FROM Учителі, Викладання
WHERE клас="11А" AND Учителі.паспорт =
      Викладання.учитель
```



## Підзапити

У попередньому розділі ми згадували про властивість реляційної замкненості мови SQL. Вона полягає в тому, що результатом будь-якого запиту є таблиця, а отже, одні запити можна підставляти в інші замість таблиць. Загалом є два способи підстановки одного запиту в інший:

- ◆ замість імені таблиці в запиті вказують ім'я іншого запиту;
- ◆ в одному запиті записують повний текст іншого запиту, взятий у круглі дужки.

Підстановку можна здійснювати у фразах FROM та WHERE. Особливо важливий і цікавий той випадок, коли у фразі WHERE одного запиту записують повний текст іншого запиту. Розглянемо цей випадок детально, реалізуювши запит *визначити прізвища та імена вчителів, що викладають в 11А класі*, з використанням підзапиту. SQL-текст запиту буде таким:

```
SELECT прізвище, [ім'я]
FROM Учителі
WHERE паспорт IN (SELECT учитель FROM Викладання
                  WHERE клас="11А")
```

У фразі WHERE цього запиту вжито оператор IN (англ. «в», «належить»). Його загальний формат такий:

значення **IN** (підзапит)

Результатом підзапиту має бути таблиця з одним стовпцем. Якщо вказане перед словом IN значення в цьому стовпці є, то весь вираз значення **IN** (підзапит) вважається істинним, інакше — хибним. Щоб краще зрозуміти принцип дії оператора IN, розглянемо крок за кроком виконання запиту *визначити прізвища та імена вчителів, що викладають в 11А класі* з точки зору СКБД.

1. Виконуємо фразу FROM. Оскільки в ній вказано лише одну таблицю Учителі, то на першому кроці ми просто «беремо» цю таблицю для подальших операцій.

2. Перебираємо всі записи таблиці Учителі, перевіряючи для кожного, чи виконується умова, вказана після слова WHERE. Ця перевірка здійснюється в два етапи.

- 2а. Виконуємо підзапит. Його результат — це набір номерів паспортів учителів 11А класу (згадайте структуру таблиці Викладання).

- 2б. Перевіряємо, чи належить значення поля паспорт з поточного запису таблиці Учителі набору значень, отриманому на кроці 2а. Якщо належить, то запис у таблиці Учителі залишаємо, інакше — відкидаємо.

3. Проектуємо отриману на кроці 2 таблицю за полями прізвище та ім'я, тобто ці поля залишаємо, а всі інші — відкидаємо.

Таким чином, оператор IN дає змогу перевірити, чи належить значення множині результатів підзапиту. Зазначимо, що у фразу WHERE підзапит можна вставляти лише за допомогою логічного оператора, формуючи вираз, значення якого істинне або хибне. Однак це не обов'язково має бути оператор IN; є ще оператори EXISTS (англ. «існує»), ANY (англ. «будь-який»), ALL (усі). Так, значення виразу EXISTS (підзапит) буде істинним, якщо результат підзапиту містить хоча б один запис, і хибним, якщо результат підзапиту порожній.

## Завдання 9.4

Використовуючи підзапит в операторі IN, реалізуйте мовою SQL запит *визначити класи, у яких викладають вчителі на ім'я Петро*.

## Віднімання множин записів

Можливості мови SQL значно ширші за можливості конструктора запитів MS Access і тому далеко не кожен SQL-запит може бути створений у вікні конструктора. Зараз ми розглянемо найпростіший різновид запитів, які не можуть бути створені за допомогою конструктора. Це запити з запереченням, наприклад *визначити прізвища та імена вчителів, які не викладають в 11А класі*. Цей запит відрізняється від того, який ми

розглядали раніше, наявністю частки «не». Якби її не було, ми би просто з'єднали таблицю вчителів із таблицею викладання, відібрали ті записи, які відповідають 10А класу, та спроектували результат на поля прізвище та ім'я. Але в запиті із часткою «не» нам потрібні не ці, а якраз всі інші вчителі. Тобто від множини всіх учителів нам потрібно відняти тих, які викладають у 10А класі. Найлегше це зробити за допомогою оператора NOT IN (англ. «не належить»):

```
SELECT прізвище, [ім'я]
FROM Учителі
WHERE паспорт NOT IN (SELECT учитель FROM Викладання
                       WHERE клас="11А")
```

Буквально запит читається так: «вибрати прізвища та імена тих учителів, які не належать множині вчителів, що викладають в 11А класі».

## Завдання 9.5

Реалізуйте мовою SQL запит *визначити назви класів, у яких не вчиться жодної дівчини*.

## Висновки

- ◆ Вибірковий запит у мові SQL має такий формат:

```
SELECT список полів, значення яких потрібно отримати
FROM список таблиць, з яких вибираються дані
WHERE умова, яку мають задовольняти записи,
      що вибираються
```

- ◆ Запит **SELECT** *x* **FROM** *y* **WHERE** *z* слід інтерпретувати так: «вибрати поля *x* тих записів таблиці *y*, які задовольняють умову *z*».
- ◆ Оператор **INNER JOIN** *b* **ON** *c* використовується у фразі **FROM** і виконує з'єднання таблиць. Він «зчіплює» ті записи таблиць *a* і *b*, які, разом узяті, відповідають умові *c*.
- ◆ Оператор **IN** дає змогу сконструювати у фразі **WHERE** умову з підзапитом. Вираз значення **IN** (підзапит) вважається

істинним, якщо значення належить множині значень, отриманій у результаті виконання підзапиту, та хибним в іншому випадку.

- ◆ Запити, у яких перед дієсловом розташовано частку «не», реалізують за допомогою оператора **NOT IN** у фразі **WHERE**. Вираз значення **NOT IN** (підзапит) вважається істинним, якщо значення не належить множині значень, отриманій у результаті виконання підзапиту, та хибним в іншому випадку.

## Завдання для самостійного виконання

Реалізуйте мовою SQL такі запити.

1. Відобразити всю інформацію про учнів, яких навчають учителі-чоловіки.
2. Для кожного вчителя відобразити всю інформацію про учнів, яких він навчає.
3. Відобразити прізвища та імена всіх учнів, крім тих, які вчаться в 11 класах.
4. Визначити прізвища та імена учнів, яких не навчає Василь Петрович Корбут.
5. Визначити прізвища вчителів, які викладають не тільки в 11 класах (зауважте, що вони можуть не викладати в 11 класах взагалі).
- 6\*. Для кожного вчителя відобразити всю інформацію про учнів, яких він не навчає.
- 7\*. Визначити прізвища вчителів, які викладають принаймні в одному тому ж класі, що і Сошко Катерина Миколаївна.
- 8\*. Визначити пари прізвищ учителів, які викладають тільки в різних класах (тобто вчитель 1 не викладає в жодному з тих класів, де викладає вчитель 2, і навпаки).

## Питання для роздумів

1. Нижче наведено формулювання та реалізації чотирьох запитів. Визначте, яка реалізація якому запиту відповідає.

## Формулювання

- I. Визначити прізвища вчителів, які викладають принаймні в одному 11 класі.
- II. Визначити прізвища вчителів, які викладають в усіх 11 класах.
- III. Визначити прізвища вчителів, які викладають не в усіх 11 класах.
- IV. Визначити прізвища вчителів, які не викладають принаймні в одному 11 класі.

## Реалізації

```
SELECT прізвище --А
FROM Учителі
WHERE NOT EXISTS (SELECT * FROM Викладання
WHERE клас Like "*11" AND
клас NOT IN (SELECT клас
FROM Викладання
WHERE учитель=
Учителі.паспорт))
```

```
SELECT прізвище --Б
FROM Учителі
WHERE EXISTS (SELECT * FROM Викладання
WHERE клас Like "*11" AND
клас NOT IN (SELECT клас
FROM Викладання
WHERE учитель=
Учителі.паспорт))
```

```
SELECT прізвище --В
FROM Учителі
WHERE EXISTS (SELECT * FROM Викладання
WHERE клас Like "*11" AND
клас IN (SELECT клас
FROM Викладання
WHERE учитель=
Учителі.паспорт))
```

```
SELECT прізвище --Г
FROM Учителі
WHERE NOT EXISTS (SELECT * FROM Викладання
WHERE клас Like "*11" AND
клас IN (SELECT клас
FROM Викладання
WHERE учитель=
Учителі.паспорт))
```

2. Припустимо, що предметною областю бази даних є механічні деталі. База містить лише одну таблицю, у якій є два поля: № деталі та № складової. У цій таблиці зберігаються відомості про те, які деталі з яких складаються. Наприклад, з поданої таблиці видно, що деталь 1 складається з деталей 2 і 3, а деталь 3 — з деталей 4 і 5.

№ деталі	№ складової
1	2
1	3
3	4
3	5

Сформулюйте для цієї бази даних запити, які:

- а) неможливо реалізувати за допомогою конструктора запитів;
- б\*) неможливо реалізувати засобами мови SQL.

## Завдання для досліджень

1. Реалізуйте завдання 9.3 та завдання для самостійного виконання 1–2 мовою SQL, але без оператора INNER JOIN.
- 2\*. З'ясуйте, який синтаксис має та як використовується оператор EXISTS. Реалізуйте за допомогою операторів EXISTS і NOT IN запит *визначити назви класів, у яких принаймні один вчитель не викладає*.
- 3\*. Реалізуйте за допомогою операторів NOT EXISTS і NOT IN запит *визначити назви класів, у яких викладають усі вчителі*.

## Розділ 10

# Групування даних

### Повторення

- ◆ Що означає «згрупувати рядки електронної таблиці за значеннями певного параметра»?
- ◆ Які операції можна виконувати над згрупованими рядками електронної таблиці?
- ◆ Опишіть алгоритм, за яким виконується функція DSUM (рос. БДСУММ) табличного процесора?
- ◆ Опишіть загальну структуру вибіркового запиту SQL.

На попередньому занятті ми вибирали в таблицях записи, що задовольняють певні критерії. Але в багатьох випадках вибрати записи — значить зробити тільки половину справи, адже часто з ними потрібно виконати якісь підсумкові операції: підрахувати їх кількість, підсумувати значення того чи іншого поля тощо. Підсумкову операцію, наприклад обчислення середнього віку учнів у кожному класі або визначення кількості класів, у яких викладає кожен учитель, часто виконують відразу над багатьма *групами записів*. У табличному процесорі для виконання таких завдань, нагадаємо, використовують проміжні підсумки та зведені таблиці, а в СКБД застосовують спеціальні різновиди запитів. Крім того, у СКБД Microsoft Access засобом відображення на аркушах друкованого формату підсумкових характеристик для груп записів є *звіти*. Ці засоби ми розглянемо на сьогоднішньому уроці.

## Групові операції в запитах

Уважно розгляньте таблицю 10.1. Це таблиця *Учні*, відсортована за значеннями поля *клас*. Один із результатів сортування полягає в тому, що записи учнів, які навчаються в одному класі, розташовано поруч. Інакше кажучи, таблицю *згруповано* за полем *клас*: спочатку розташовано групу учнів 10А класу, потім — 10Б, 11А, і нарешті — 11Б класу.

код	прізвище	ім'я	дата народження	стать	клас
1	Шпак	Максим	11.12.1996	ч	10А
5	Григорук	Петро	05.05.1997	ч	10Б
6	Райчук	Олена	12.01.1998	ж	10Б
2	Хоменко	Ірина	09.09.1995	ж	11А
3	Недригайло	Петро	01.07.1995	ч	11Б
4	Буджак	Оксана	14.01.1996	ж	11Б

Табл. 10.1. Таблиця *Учні*, відсортована за полем *клас*

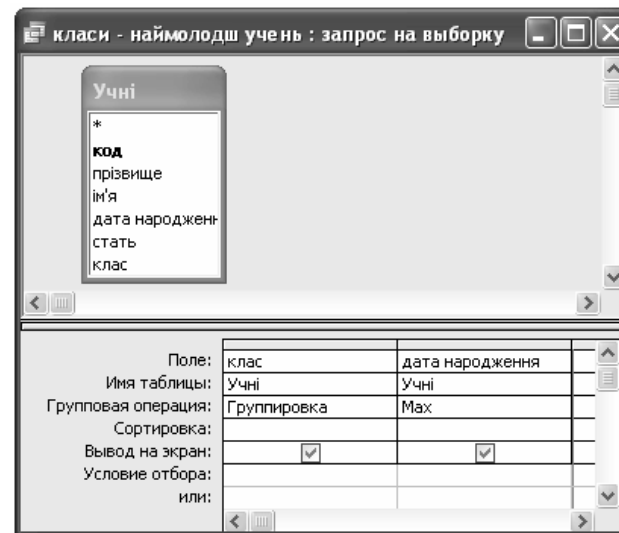
У цій таблиці легко обчислювати підсумкові показники для груп записів. Наприклад, у табл. 10.2 показано результат обчислення для кожного класу кількості учнів і дати народження наймолодшого з них.

Групування шляхом сортування ми виконували в табличному процесорі Microsoft Excel. У СКБД Microsoft Access для групування записів сортувати таблицю не потрібно. Групування і обчислення підсумкових операцій здійснюють за допомогою конструктора запитів. У вікні конструктора потрібно клацнути правою кнопкою миші поле, за значеннями якого будуть виділятися групи, і вибрати у контекстному меню команду **Групповые операции** (Групові операції). Після цього в бланк запиту буде додано рядок **Групповая операция** (рис. 10.1, а).

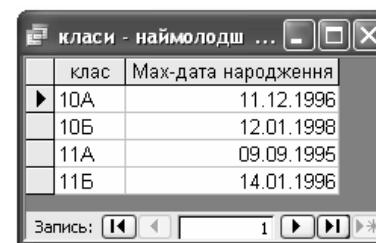
код	прізвище	ім'я	дата народження	стать	клас
1	Шпак	Максим	11.12.1996	ч	10А
<b>Дата народження наймолодшого учня</b>			<b>11.12.1996</b>		
<b>Кількість учнів</b>	<b>1</b>				
5	Григорук	Петро	05.05.1997	ч	10Б
6	Райчук	Олена	12.01.1998	ж	10Б
<b>Дата народження наймолодшого учня</b>			<b>12.01.1998</b>		
<b>Кількість учнів</b>	<b>2</b>				
2	Хоменко	Ірина	09.09.1995	ж	11А
<b>Дата народження наймолодшого учня</b>			<b>09.09.1995</b>		
<b>Кількість учнів</b>	<b>1</b>				
3	Недригайло	Петро	01.07.1995	ч	11Б
4	Буджак	Оксана	14.01.1996	ж	11Б
<b>Дата народження наймолодшого учня</b>			<b>14.01.1996</b>		
<b>Кількість учнів</b>	<b>2</b>				

Табл. 10.2. Обчислення підсумкових показників для груп записів у табличному процесорі

З рис. 10.1, б видно, що результати групування рядків таблиці в табличному процесорі та в СКБД дещо відмінні. У СКБД результатом запиту з групуванням є таблиця, де кожна група записів початкової таблиці «згортається» в один запис (наприклад, на рис. 10.1, б замість двох записів «10Б» ми бачимо один). Власне об'єднання кількох записів в один в реляційних базах даних і називається групуванням.



а



б

Рис. 10.1. Групування таблиці: а — бланк запиту; б — результат виконання запиту

**Група** — це набір записів з однаковим значенням певного поля. Під групуванням розуміють процес об'єднання групи записів в один; поле, значення якого однакові, називається **полем групування**.

Однак із самого факту згортання кількох записів в один мало зиску. Групування стає справді корисним, якщо до груп записів застосовують певну підсумкову операцію. Найуживанішими підсумковими операціями вважаються:

- ◆ визначення кількості записів (Count);
- ◆ обчислення сумарного значення (Sum);
- ◆ обчислення середнього значення (Avg);
- ◆ обчислення максимуму (Max);
- ◆ обчислення мінімуму (Min).

Усі операції, крім визначення кількості записів, виконують над значеннями певного *поля підсумків*. Зауважте, що це не поле групування, а якесь інше поле числового типу, грошового типу або типу дата/час. На рис. 10.1 полем групування є клас, полем підсумків — дата народження, а підсумковою операцією — обчислення максимуму (у наймолодшого учня дата народження максимальна). Тип підсумкової операції вибирають зі списку, що розкривається кнопкою ▼, розташованою в клітинках рядка **Групповая операция** справа.

Зазначимо, що коли підсумкова операція полягає у визначенні кількості записів у групі, то в конструкторі запитів ми також маємо вибрати певне поле підсумків, але неважливо, яке саме. Справді, кількість рядків не залежить від того, за яким стовпцем ми її підраховуємо.

**Для допитливих.** Як полів групування, так і полів підсумків у запиті може бути кілька. Якщо використовують кілька полів групування, то до однієї групи відносять записи, які в усіх цих полях мають однакові значення.

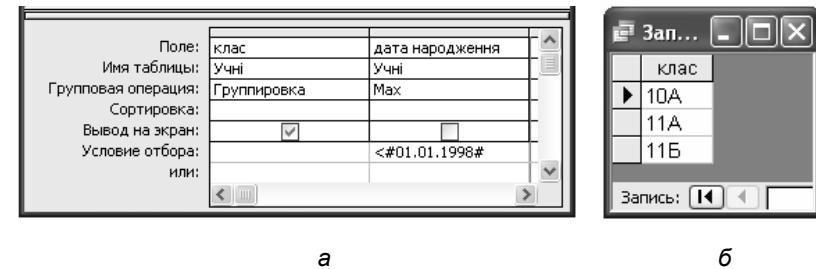
## Завдання 10.1

У базі даних **школа** створіть і виконайте запит *визначити дату народження наймолодшого учня в кожному класі*.

Поле підсумків можна використовувати в умовах відбору так само, як і будь-яке інше поле. Самі умови записують у рядку **Условие отбора** конструктора запитів. Як приклад розглянемо

подані на рис. 10.2 бланк і результат виконання запити «Визначити класи, всі учні яких народилися до 1 січня 1998 року». СКБД виконує запит за таким алгоритмом.

1. Таблиця учнів групується: до однієї групи потрапляють записи учнів, які навчаються в одному класі.
2. Для кожної групи обчислюється максимальна дата народження учня.
3. Для кожної групи створюється один запис у таблиці результатів. У цьому записі два значення: назва класу та максимальна дата народження його учнів.
4. У таблиці результатів залишаються тільки ті записи, для яких максимальна дата народження менша за 1 січня 1998 року.



**Рис. 10.2.** Запит «Визначити класи, всі учні яких народилися до 1 січня 1998 року»: а — бланк запити; б — результат виконання

Запити з групуванням можна створювати не за однією, а за кількома таблицями, з'єднуючи їх так само, як і в запитах без групування. Приклади кількатабличних запитів із групуванням буде розглянуто далі.

**Для допитливих.** У запитах із групуванням можна вказувати лише поля групування та підсумків, адже тільки вони мають одне значення для кожної групи і тому можуть бути подані у вигляді одного запису. «Звичайного» поля не може бути, оскільки в кожній групі його значень багато і незрозуміло, яке з них відобразити в таблиці результатів.

## Завдання 10.2

Створіть і виконайте запит *визначити класи, всі учні яких народилися до 1 січня 1998 року*.

### Приклади запитів

#### Приклад 1

*Для кожного вчителя визначити, скількох учнів він навчає.*

Вікно конструктора запиту зображено на рис. 10.3. Прізвища та паспорти вчителів беремо з таблиці *Учителі*, яку з'єднано з таблицею *Викладання*, її — з таблицею *Класи*, а вже таблицю *Класи* — з таблицею *Учні*. Зауважте, що прямий зв'язок між таблицею *Класи* та вчителів видалено, оскільки він визначає, хто є класним керівником, а це за умовою запиту не потрібно. Полями групування є паспорт та прізвище вчителя. Поле підсумків може бути будь-яким, оскільки підсумкова операція — обчислення кількості записів.

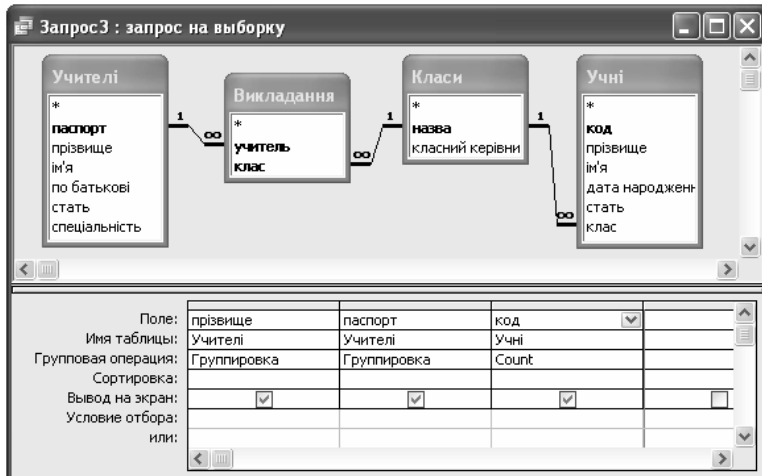


Рис. 10.3. Вікно конструктора запиту «Для кожного вчителя визначити, скількох учнів він навчає»

#### Приклад 2

Припустимо, що до таблиці *учнів* додано поле *успішність*, у якому зберігається середній бал в таблиці за минулий навчальний рік. Запит є таким: *визначити прізвища класних керівників тих класів, середня успішність у яких нижча за 8 балів*.

Вікно конструктора запиту зображено на рис. 10.4. У запиті два поля групування: прізвище вчителя та назва класу. Групувати записи за назвою класу потрібно тому, що саме для всіх учнів кожного класу обчислюється середня успішність, а за прізвищем вчителя — тому, що значення цього поля відображається. Пояснимо друге твердження. У запиті з групуванням, нагадаємо, будь-яке поле має бути полем групування або підсумків. Прізвище вчителя — це не підсумкове значення, а отже, відповідне поле має бути полем групування, а не підсумків.

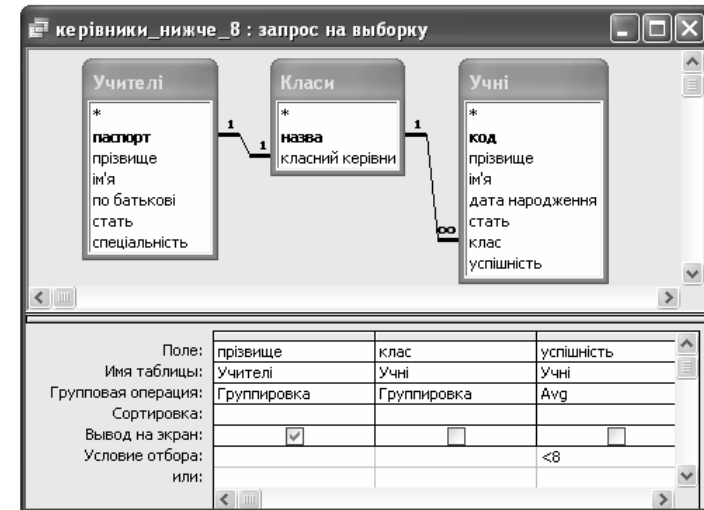


Рис. 10.4. Вікно конструктора запиту «Визначити прізвища класних керівників тих класів, середня успішність у яких нижча за 8 балів»

Зауважте також, що в цьому запиті на результат підсумкової операції накладено умову відбору (середня успішність нижча 8 балів). Зрозуміло, що ця умова перевірятиметься для вже згру-

пованої таблиці, тобто відповідати чи не відповідати їй будуть цілі класи.

### Приклад 3

*Визначити середню успішність учнів, яких навчає Сошко Катерина.*

Вікно конструктора запиту зображено на рис. 10.5. У цьому запиті умову відбору (учитель — Сошко Катерина) накладено на поле групування. Можна вважати, що запит виконується так: після з'єднання таблиць відбираються записи, що стосуються Сошко Катерини, а потім обчислюється середнє значення поля успішність в отриманій таблиці.

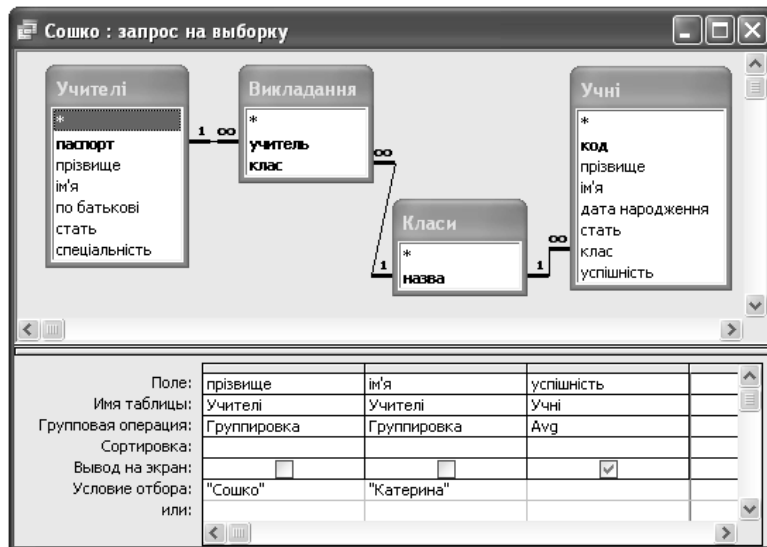


Рис. 10.5. Вікно конструктора запиту «визначити середню успішність учнів, яких навчає Сошко Катерина»

### Завдання 10.3

Реалізуйте запити, розглянуті в прикладах 1–3. Для створення запитів з прикладів 2 і 3 додайте до таблиці Учні поле успішність.

### Завдання 10.4

Реалізуйте такі запити в базі даних школа (до таблиці учнів додано поле успішність).

- Визначити найвищу успішність учнів 11А класу.
- Для кожного класу визначити, скільки математиків у ньому викладає.
- Визначити, у скількох учнів, молодших за 16 років, викладає Петрова Ніна Володимирівна. (Подумайте, яке обмеження потрібно накласти на дату народження учня.)

### Групування в мові SQL\*

Відобразіть запит, створений у завданні 10.1 (визначити дату народження наймолодшого учня в кожному класі), у режимі SQL. Його текст має бути таким.

```
SELECT Учні.клас, Max(Учні.[дата народження]) AS  
[Max-дата народження]  
FROM Учні  
GROUP BY Учні.клас;
```

Нове для вас ключове слово — GROUP BY, за допомогою якого, власне, і реалізують групування. Назви полів групування перелічують після цього слова, розділяючи комами, якщо таких полів кілька. Фразу GROUP BY записують після фрази WHERE або після фрази FROM, якщо фраза WHERE відсутня.

Звернімо увагу на особливості фрази SELECT запиту з групуванням.

- У цій фразі вказують хоча б одне поле групування. (Якщо жодного поля групування у фразі SELECT не вказати, у результаті запиту ми не побачимо груп і запит стане безглуздим.)
- Крім поля групування у фразі SELECT можна зазначити підсумкові функції, перелічені вище. Аргумент такої функції — назва поля підсумків, яку записують у дужках. Наприклад, у нашому запиті функція називається Max, а її аргумент — Учні.[дата народження].



3. Полю, де відображаються підсумки, можна надати назву, записавши її після ключового слова AS. Зазначимо, що за допомогою цього слова можна перейменувати будь-які поля в таблиці результатів запиту.
4. Крім полів групування та підсумкових функцій у фразі SELECT запиту з групуванням неможна вказувати жодних інших полів. Щоб зрозуміти, чому так, припустимо, що в розглянутому щойно запиті ми відображуємо ще прізвище учня:

```
SELECT Учні.клас, Учні.прізвище, ...
...
GROUP BY Учні.клас
```

Оскільки для кожної групи (у даному випадку — класу) в таблиці результатів відводиться один рядок (див. рис. 10.1, б), а в класі може бути кілька учнів, то їх прізвища просто не вмістяться в одній клітинці (нагадаємо, що кожна клітинка реляційної БД містить тільки одне значення). Саме тому така конструкція неприпустима.

### Завдання 10.5\*

Реалізуйте мовою SQL запит *визначити, скільки учнів навчається в кожному класі*. Для обчислення кількості використайте функцію Count. Її аргументом може бути \* (символ, що позначає будь-яке поле), тому що, як зазначалося вище, неважливо, за яким полем рахувати кількість записів.

Тепер розглянемо SQL-текст запиту, створеного у завданні 10.2: визначити класи, всі учні яких народилися до 1 січня 1998 року.

```
SELECT Учні.клас
FROM Учні
GROUP BY Учні.клас
HAVING Max(Учні.[дата народження])<#1/1/1998#;
```

У ньому є нове для вас ключове слово HAVING. Після цього слова записують умову відбору груп записів, тобто фраза HAVING відіграє для груп записів ту саму роль, що і фраза WHERE для окремих записів. У цілому наведений запит СКБД виконує так:

1. Таблиця Учні групується за полем клас: до однієї групи відносяться записи учнів одного класу.
2. Для кожної групи перевіряється умова Max(Учні.[дата народження])<#1/1/1998#, тобто чи народився наймолодший учень до 1 січня 1998 року.
3. Значення поля клас тих груп, для яких умова виконується, відображується в таблиці результатів.

### Завдання 10.6\*

Реалізуйте мовою SQL запит *визначити прізвища вчителів, які викладають більше, ніж в одному класі*.

### Звіти

Підіб'ємо певні підсумки. У реляційній базі даних можна створювати запити двох різновидів: із групуванням та без нього. У запитах із групуванням усі записи в кожній групі об'єднуються в один, а результат запиту без групування — це таблиця, у якій відображено значення всіх полів усіх записів (рис. 10.6, а). Однак у деяких випадках для користувача незручна ні перша, ні друга форма подання інформації. Скажімо, у запиті без групування «Для кожного вчителя визначити прізвища та імена учнів, яких він навчає», результат виконання якого зображено на рис. 10.6, а, доречно було б виконати «м'яке» групування, відобразивши по одному разу прізвище, ім'я та по батькові кожного вчителя, а під ним — список усіх його учнів (рис. 10.6, б). У Microsoft Access подати дані в такий спосіб дозволяють *звіти* (рос. *Отчеты*). Звіти — це екранні форми, що призначені насамперед для подальшого друку на папері, і тому найчастіше вони подаються у вигляді аркушів паперу формату А4.

Учителі.прізвищ	Учителі.ім'я	Учні.прізвище	Учні.ім'я
Сошко	Катерина	Шлак	Максим
Петрова	Ніна	Шлак	Максим
Сошко	Катерина	Хоменко	Ірина
Корбут	Василь	Хоменко	Ірина
Петрова	Ніна	Хоменко	Ірина
Михайлюк	Дмитро	Недригайло	Петро
Михайлюк	Дмитро	Буджак	Оксана
Корбут	Василь	Недригайло	Петро
Корбут	Василь	Буджак	Оксана
Петрова	Ніна	Недригайло	Петро
Петрова	Ніна	Буджак	Оксана

а

### Учителі та учні

Учителі.прізвище	Учителі.ім'я	Учні.прізвище	Учні.ім'я
Корбут	Василь	Буджак	Оксана
		Недригайло	Петро
		Хоменко	Ірина
Михайлюк	Дмитро	Буджак	Оксана
		Недригайло	Петро
		Хоменко	Ірина
Петрова	Ніна	Буджак	Оксана
		Недригайло	Петро
		Хоменко	Ірина
Сошко	Катерина	Шлак	Максим
		Хоменко	Ірина
		Шлак	Максим

б

Рис. 10.6. Подання результатів запити: а — в табличній формі; б — у формі звіту

В одному звіті може відображатися інформація з однієї або кількох таблиць або запитів. Створювати звіти, як і інші об'єкти БД, можна в режимі майстра або за допомогою конструктора. Зауважимо, що коли у звіті відображаються дані з кількох таблиць, краще спочатку створити запит, де ці таблиці з'єднуються, а вже потім, на основі цього запиту, — звіт.

Подібно до форм, звіти найзручніше створювати в режимі майстра. Розглянемо детально всі кроки майстра на прикладі звіту за таким запитом.

*Для кожного вчителя визначити класи, у яких він викладає, а також прізвища, імена та середню успішність учнів цих класів.*

### Попередній крок

Створюємо запит, умову якого ми щойно сформулювали. Вікно конструктора запиту показано на рис. 10.7. Зауважте, що прями зв'язок між таблицею класів та вчителів видалено, оскільки він визначає, хто є класним керівником, а за умовою запиту нам така інформація не потрібна. Зберігаємо запит під назвою Учителі\_класи\_учні.

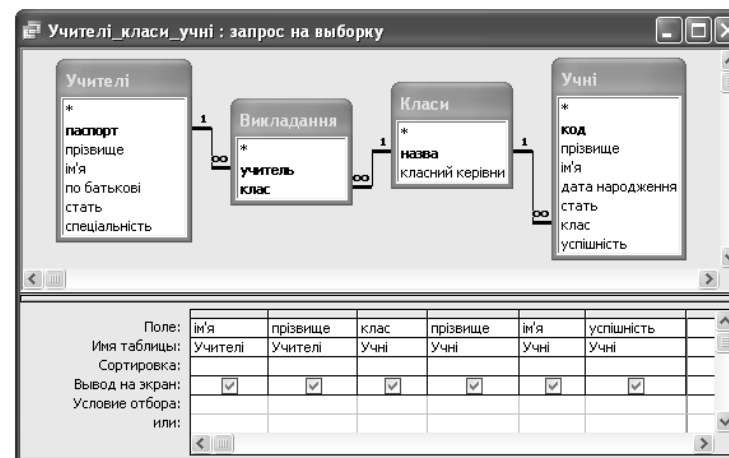


Рис. 10.7. Вікно конструктора запиту «Для кожного вчителя визначити класи, у яких він викладає, а також прізвища, імена та середню успішність учнів цих класів»

## Крок 1

Запускаємо майстер звітів, клацнувши кнопку **Создание отчета с помощью мастера** (Створення звіту за допомогою майстра) на вкладці **Отчеты** головного вікна бази даних. Перший крок майстра звітів такий самий, як і майстрів форм та запитів. На ньому потрібно вибрати поля, дані з яких відобразатимуться у звіті. У нашому випадку зі списку **Таблицы и запросы** слід вибрати запит `Учителі_класи_учні`, а потім за допомогою кнопки **»** — усі його поля (рис. 10.8).

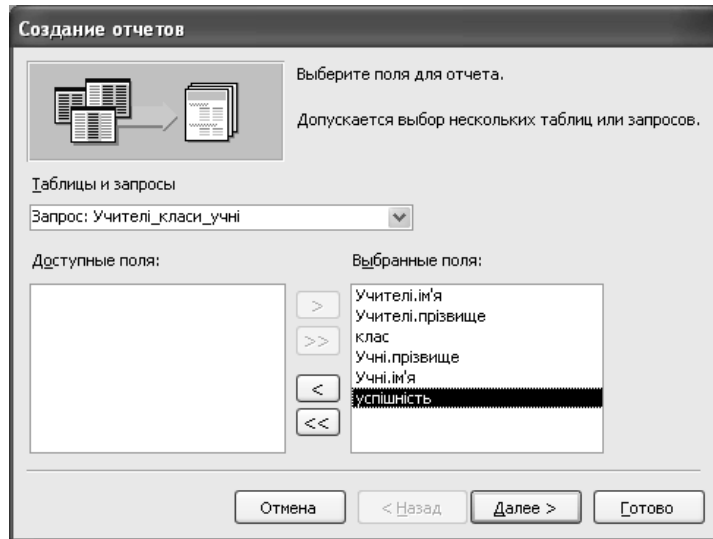


Рис. 10.8. Визначення полів, що відобразатимуться у звіті

## Крок 2

На другому кроці визначаємо, яка таблиця буде головною, а яка — підлеглою, як ми це робили під час побудови форми за кількома таблицями. Головною таблицею є та, за значеннями полів якої проводитиметься групування, а підлеглою — та, значення полів якої відобразатимуться в групах. У нашому звіті головною буде таблиця вчителів, а підлеглою — таблиця учнів, оскільки для кожного вчителя ми виділятимемо групу учнів (рис. 10.9).

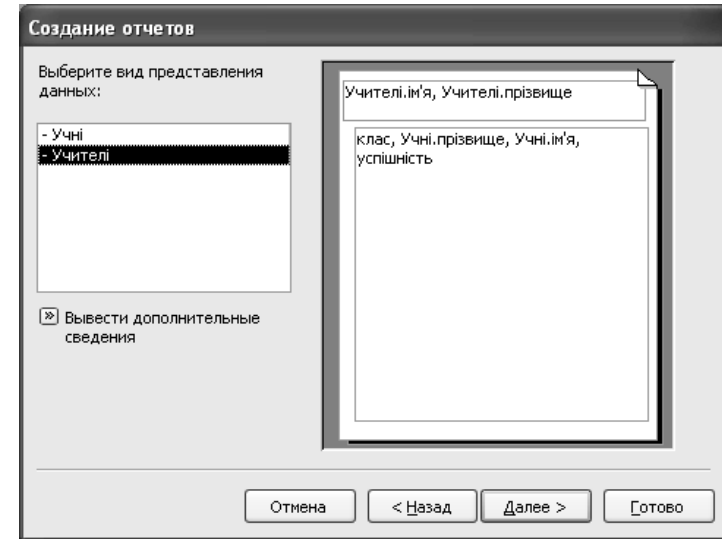


Рис. 10.9. Визначення головної та підлеглої таблиць

## Крок 3

На третьому кроці майстра можна задати поля, за якими виконуватиметься додаткове багаторівневе групування. У нашому прикладі групування першого рівня здійснюється за вчителями, а учнів, які відповідають одному вчителю, можна ще поділити за класами — це буде групування другого рівня (рис. 10.10). Додаткові поля для групування вибирають за допомогою кнопки **>**, а рівень групи змінюють за допомогою кнопок **▲** та **▼**.

## Крок 4

На цьому кроці визначають поля, значення яких сортуватимуться. Крім того, якщо серед полів звіту є числові, грошові або поля дати/часу, то у вікні майстра з'являється кнопка **Итоги** (Підсумки), за допомогою якої можна задати підсумкову операцію для таких полів. Наприклад, у нашому запиті можна обчислити середню успішність учнів кожного класу (рис. 10.11). Залежно від того, який перемикач встановлено у групі **Показать**, відобразатимуться або тільки результати підсумкових операцій (**только итоги**), або і підсумкові значення, і самі дані (**данные и итоги**).

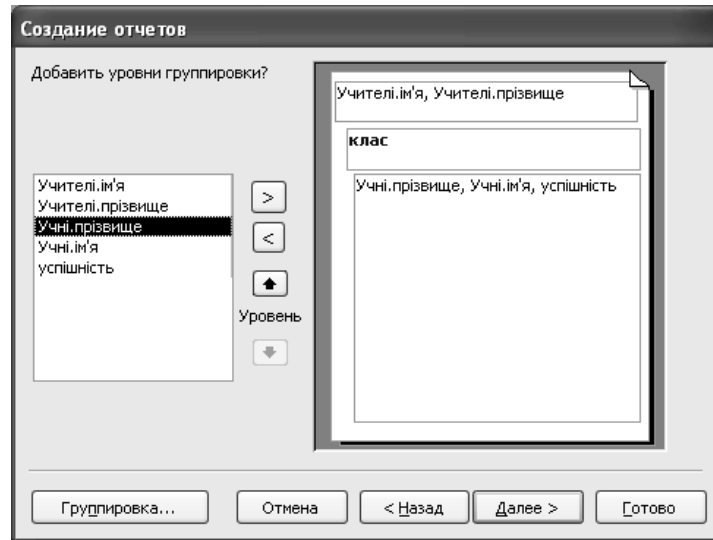


Рис. 10.10. Додаткове групування у звітах

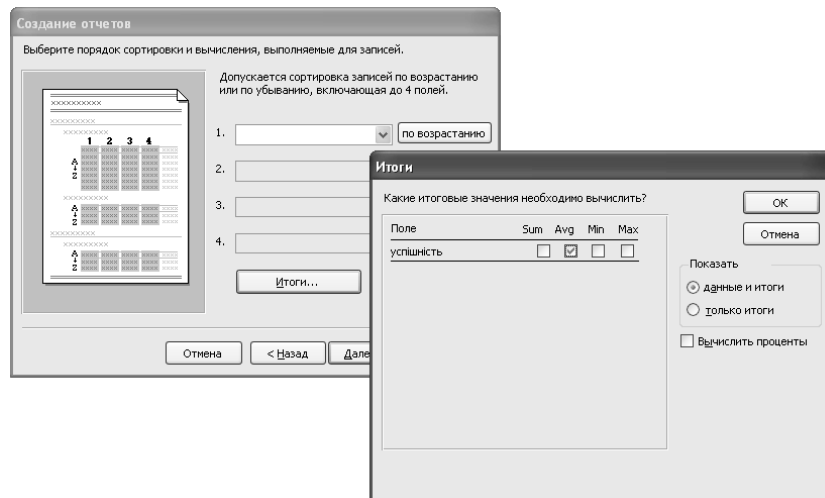


Рис. 10.11. Виконання підсумкових операцій у звітах

## Кроки 5–7

На п'ятому кроці вибирають спосіб розташування та вирівнювання даних у звіті. Якщо має відображатися багато полів, краще обрати альбомну орієнтацію аркуша. На шостому кроці слід задати стиль оформлення звіту, а на сьомому — його ім'я.

Готовий звіт (рис. 10.12) можна переглянути, двічі клацнувши його значок, та відредагувати за допомогою конструктора. Для цього звіт потрібно виділити в головному вікні бази даних і клацнути кнопку **Конструктор**, щоб перейти у вікно конструктора звітів. Найчастіше режим конструктора використовують для розширення і редагування написів.

Ім'я учителя	Прізвище учителя	Клас	Прізвище учня	Ім'я учня	Успішність
Василь	Корбут	11А	Хоменко	Ірина	10,6
Середня успішність класу					<b>10,60</b>
		11Б	Буджак	Оксана	11
			Недригайло	Петро	9
Середня успішність класу					<b>10</b>
Середня успішність в сіх учнів учителя					<b>10,2</b>

Рис. 10.12. Визначення способу розташування й вирівнювання даних у звіті

## Завдання 10.7

Сконструйте звіт за запитом для кожного вчителя визначити класи, у яких він викладає, та прізвища й імена учнів, яких

він навчає, виконавши кроки, описані вище. Відобразіть також середню успішність учнів кожного класу.

**Для допитливих.** Якщо ви побудували звіт на основі якогось запиту, а потім цей запит змінили, на звіті ці зміни ніяк не відіб'ються. Щоб дійсно змінити набір даних, які відображаються у звіті, потрібно клацнути правою кнопкою миші індикатор  в лівому верхньому куті вікна конструктора звіту, вибрати з контекстного меню команду **Свойства** і на вкладці **Данные** вікна властивостей звіту клацнути кнопку , розташовану справа від поля **Источник записей**. У результаті ви перейдете знайоме вікно конструктора запиту. Але це буде той запит, який справді пов'язаний зі звітом (в області запитів головного вікна бази даних його немає).

## Завдання 10.8

Створіть звіт, у якому для кожного класу відобразатимуться прізвища, імена та спеціальності вчителів, що в ньому викладають, а також кількість таких учителів.

## Висновки

- ◆ Група — це набір записів з однаковим значенням певного поля. Під групуванням розуміють процес об'єднання групи записів в один; поле, значення якого однакові, називається полем групування.
- ◆ Підсумкова операція полягає в обчисленні в кожній групі записів певної підсумкової характеристики, наприклад, максимуму, суми, середнього. Підсумкову операцію виконують над значеннями якогось поля, що називається полем підсумків.
- ◆ Поле підсумків повинно мати числовий тип, грошовий тип або тип дати/часу і бути не тим полем, за яким здійснювалося групування.
- ◆ Операцію визначення кількості записів можна виконувати над значеннями будь-якого поля; її результат від вибору поля не залежить.

- ◆ Загальний синтаксис SQL-запиту з групуванням такий:

**SELECT** список полів, значення яких потрібно отримати

**FROM** список таблиць, з яких вибираються дані

**WHERE** умова, яку мають задовольняти записи, що вибираються

**GROUP BY** список полів групування

**HAVING** умова, яку мають задовольняти групи записів

- ◆ Звіти — це екранні форми, що подаються у вигляді аркушів паперу і призначені насамперед для подальшого друку. За допомогою звітів можна візуально групувати записи, не об'єднуючи їх.

## Завдання для самостійного виконання

1. Реалізуйте такі запити.

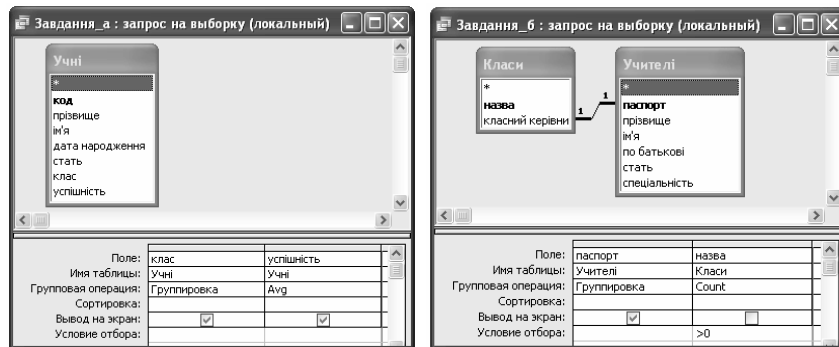
- а) Визначити назви класів, у яких навчається більше 2 учнів.
- б) Визначити прізвища вчителів, які викладають більш, ніж в одному класі.
- в) Визначити дату народження наймолодшого учня.
- г) Визначити кількість учителів, що є класними керівниками.
- д) Для кожного вчителя визначити максимальну успішність його учнів (використайте поле успішність).
- е\*) Визначити прізвище наймолодшого учня.
- є\*) Визначити спеціальності, за якими працюють тільки вчителі-жінки.

2. Створіть звіти за описаними далі запити. Передбачте групування, де це доречно.

- а) Для кожного класу визначити прізвища та імена учнів, що в ньому навчаються, а також їх найвищу та найнижчу успішність.
- б) Для кожної спеціальності визначити вчителів, що її мають, кількість таких учителів та класи, у яких вони викладають.

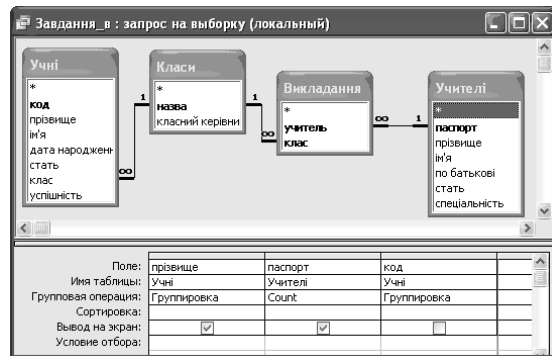
## Питання для роздумів

1. Сформулюйте запити, реалізацію яких показано на рис. 10.13, а–в.
- 2\*. Запишіть SQL-текст запитів, вікна конструктора яких зображено на рис. 10.13, а–в.



а

б



в

Рис. 10.13. Вікна конструктора запитів

- 3\*. Прочитайте уважно коментар до прикладу 3, розглянутого в цьому розділі. Зауважте, що фактично групування даних в цьому запиті не відбувається, хоча й виконується підсум-

кова операція. Спробуйте реалізувати запит з прикладу 3 мовою SQL, використавши тільки фрази SELECT, FROM і WHERE. У фразі SELECT слід застосувати агрегатну функцію.

- 4\*. Чому у запиті, розглянутому в прикладі 1, полями групування є паспорт і прізвище, а не тільки прізвище? Наведіть приклад даних, коли групування лише за полем прізвище давало б некоректний результат.
- 5\*. Припустимо, що до бази даних **школа** додано таблицю Предмет, яку з'єднано з таблицями вчителів і класів через таблицю Викладання. Чому для такої бази реалізація запиту «для кожного вчителя визначити, скількох учнів він навчає», зображена на рис. 10.3, буде некоректною?

## Завдання для досліджень

1. З'ясуйте, для чого призначений перехресний запит і як його конструювати. Додайте до бази даних **школа** таблицю Предмет з полями назва та кількість уроків на тиждень і з'єднайте її зв'язком з таблицею Викладання (визначте самостійно множинність зв'язку та створіть зовнішній ключ, потрібний для його моделювання). Побудуйте перехресний запит *визначити, скільки предметів викладає кожен учитель кожному учневі*.
2. До звіту, створеного в завданні для самостійного виконання 2б з цього розділу, додайте поле, де відобразатиметься середня успішність учнів, що навчаються у вчителів кожної спеціальності.

## Розділ 11°

# Автоматизоване видалення, оновлення і додавання даних

### Повторення

- ◆ Назвіть основні функції систем керування базами даних.
- ◆ Які операції з редагування даних можна виконувати в СКБД MS Access?
- ◆ Опишіть загальну структуру та спосіб інтерпретації вибіркового запиту SQL.
- ◆ Що таке каскадне оновлення та каскадне видалення даних?

Відкривши таблицю, ви можете не лише вводити до неї нові записи, але й змінювати або видаляти наявні. Проте в режимі редагування таблиці це зручно робити тільки тоді, коли потрібно змінити чи видалити невелику кількість записів. Наприклад, якщо якийсь учень перейшов із 10А класу до 10Б, слід відкрити таблицю *Учні*, знайти відповідний запис та ввести нове значення в поле *клас*. Але якщо в базі даних є кілька тисяч записів щодо учнів багатьох шкіл, то наприкінці навчального року, коли вони переходять у наступний клас, змінювати всі їхні записи вручну буде вкрай проблематично. Автоматизувати подібні операції дозволяє спеціальний різновид запитів — *запити на оновлення даних*. Те саме можна сказати і про додавання та видалення записів. Запити на додавання, оновлення та видалення даних ми вивчатимемо сьогодні.

## Видалення даних

Розглянемо детальніше задачу оновлення бази даних *школа* після завершення навчального року. Вона складається з кількох дрібніших підзадач, а саме:

1. Видалити записи, що стосуються одинадятикласників, оскільки вони більше не є учнями.
2. Змінити записи всіх інших учнів: у полі *клас* замість 10А записати 11А, замість 10Б — 11Б, замість 9А — 10А і т.д.
3. Змінити відомості про викладання і класне керівництво: якщо певний учитель викладав у 10А класі, тепер він, скоріш за все, викладатиме в 11А, якщо він був класним керівником 9Б, то стане керівником 10Б тощо.

Звичайно, цим списком оновлення, які потрібно виконати в базі даних наприкінці навчального року, не обмежуються. Якийсь учитель може помінятися, якийсь — звільнитися, учень може перевестися в інший клас або школу тощо. Проте зазначені зміни локальні, вони стосуються окремих записів і тому їх можна вносити вручну, в той час як три перелічених вище завдання варто виконувати автоматизовано над багатьма записами відразу.

Почнемо з автоматизованого видалення записів одинадятикласників. Його здійснюють за допомогою *запиту на видалення даних*, порядок створення та виконання якого описано у праві 11.1.

**Запит на видалення даних** призначено для видалення з таблиці записів, що задовольняють певну умову.

### Вправа 11.1

Створіть і виконайте запит *видалити з таблиці Учні записи, що стосуються одинадятикласників*.

1. Створіть копію файлу бази даних *школа*, назвавши її *школа\_кінець\_року*. Надалі працюйте з цією копією.

- Відкрийте конструктор запитів. У вікні **Добавление таблицы** виберіть таблицю **Учні**, оскільки запит стосується учнів, після чого клацніть кнопку **Добавить** і закрийте це вікно.
- Задайте тип запиту. Якщо ви працюєте в Access 2003, у меню **Запрос** виберіть команду **Удаление**, а якщо в Access 2007/2010 — клацніть кнопку **Удаление** на стрічці **Конструктор**. Після виконання цієї команди бланк запиту зміниться: замість рядків **Сортировка** та **Вывод на экран** з'явиться рядок **Удаление**.
- Уведіть у бланк запиту дані за зразком, показаним на рис. 11.1. Поле клас перетягніть з верхньої частини вікна конструктора, а умову відбору введіть уручну.

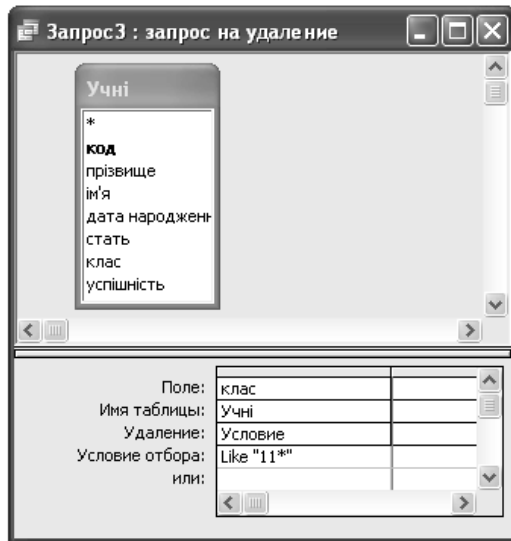



Рис. 11.1. Бланк запиту на видалення даних

- Запустіть запит на виконання кнопкою . Буде відображено вікно з проханням підтвердити видалення 3 записів. Клацніть у ньому кнопку **Да**.
- Відкрийте таблицю **Учні** і переконайтеся, що відомостей про одинадятикласників тепер у ній немає.

Пояснимо принцип дії запиту. Під час його виконання видаляються ті записи, які задовольняють умову відбору, накладену на значення одного або кількох полів. Цю умову задають у бланку запиту так само, як і в запиті на вибірку даних: у рядку **Поле** слід вказати назву поля, а в рядку **Условие отбора** — умову, яку мають задовольняти значення цього поля. У цьому запиті умова має вигляд `Like "11*"`. Як ви вже знаєте, це означає «всі значення, що починаються з символів 11», тобто значення, що відповідають одинадятикласникам.

У мові SQL запит на видалення даних має дуже просту структуру:

```
DELETE *
FROM ім'я таблиці
WHERE умова відбору записів
```

Наприклад, SQL-текст запиту, створеного у вправі 11.1, виглядатиме так:

```
DELETE *
FROM Учні
WHERE клас Like "11*"
```

Слово `DELETE` у перекладі з англійської означає «видалити», а в цілому останній запит слід читати так: «видалити з таблиці **Учні** записи, у яких значення поля **клас** починається з символів 11». Символ `*` після слова `DELETE` означає «всі поля»: коли ми видаляємо той чи інший запис, видаляються всі його поля.

## Завдання 11.1

Створіть і виконайте запити на видалення з таблиці **Викладання** всіх записів, що стосуються викладання в 11 класах, а потім — на видалення з таблиці **Класи** всіх записів про 11 класи.

Зауважте, що якби під час створення зв'язків між таблицями **Класи** та **Учні**, а також **Класи** і **Викладання** було задано режим каскадного видалення даних, видаляти всі відомості про 11 класи було б простіше. А саме, достатньо було б видалити



записи про 11 класи з таблиці Класи, а записи про одинадцятикласників і викладання в 11 класах видалилися б при цьому автоматично.

Для допитливих. У фразі WHERE запиту на видалення можна використовувати логічну зв'язку AND, оператор IN та всі інші засоби, які використовують у фразі WHERE вибіркового запиту. Це дає змогу конструювати достатньо складні умови відбору записів, що видалитимуться. Наприклад, запит *видалити відомості про вчителів, що викладають в 11 класах*, виглядатиме так:

```
DELETE *
FROM Учителі
WHERE паспорт IN (SELECT учитель FROM Викладання
                  WHERE клас Like "11*")
```

## Оновлення даних

Коли після завершення навчального року дані про одинадцяті класи видалено, учнів усіх інших класів потрібно перевести до наступного класу: учнів 9А класу — до 10А, учнів 10Б — до 11Б тощо. Це ми зробимо за допомогою *запиту на оновлення даних*, однак спочатку, щоб полегшити завдання, внесемо зміни в схему даних.

### Оновлення схеми даних

У базі даних *школа\_кінець\_року* виділимо окреме поле для номера, або паралелі класу (наприклад, 10,11) та окреме поле для букви (наприклад, А,Б). Ці зміни зображено на рис. 11.2. Зауважте, що тепер ключ таблиці класів складатиметься з двох атрибутів, які обидва потрібно використовувати у зв'язках з таблицями Викладання та Учні, відповідним чином змінивши в них зовнішні ключі. У вправі 11.2 опишемо детально, як оновити схему даних.

## Вправа 11.2

Оновіть схему даних *школа\_кінець\_року* згідно з рис. 11.2. Вам потрібно замінити поле назва у таблиці Класи двома іншими полями, створити у цій таблиці складений ключ і змінити відповідним чином її зв'язки з таблицями Викладання та Учні. Подібні дії ви вже виконували під час роботи з розділом 5.

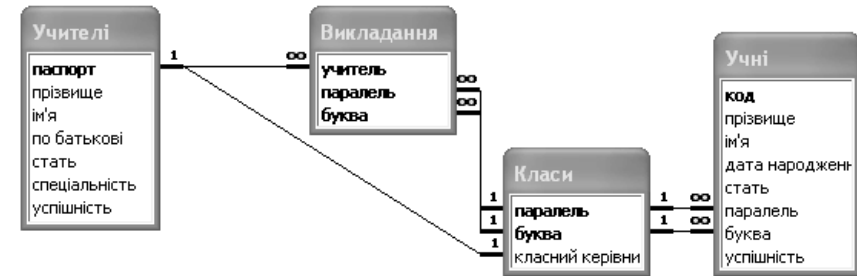




Рис. 11.2. Оновлена схема бази даних *школа*

- Відкрийте вікно **Схема даних**, скориставшись кнопкою  на панелі інструментів у Access 2003 або на стрічці **Робота с базами даних** в Access 2007/2010, та видаліть зв'язки між таблицями Класи та Учні, а також Класи та Викладання. Щоб видалити зв'язок, потрібно виділити його, клацнувши лінію зв'язку лівою кнопкою миші, а потім натиснути клавішу **Del** та підтвердити видалення.
- Змініть структуру таблиці Класи. Для цього перейдіть в режим конструктора цієї таблиці і виконайте такі дії.
  - Додайте нове поле перед полем назва, клацнувши його правою кнопкою миші й вибравши з контекстного меню команду **Добавить строки**.
  - Дайте новому полю назву паралель та виберіть тип **Числовой**, а поле назва перейменуйте на буква.
  - Створіть складений ключ з полів паралель та буква: клацніть індикатори цих полів, утримуючи клавішу **Ctrl**, а потім клацніть кнопку . На індикаторах обох полів має з'явитися позначка у вигляді ключа.

3. Додайте числові поля паралель до таблиць Викладання та Учні, а поля клас в цих таблицях перейменуйте на буква.
4. Змініть дані у таблицях Класи, Учні та Викладання, поділивши назви класів на дві частини. Наприклад, замість значення 10А в полі назва запишіть 10 у полі паралель і А в полі буква.
5. Створіть зв'язок між таблицями Класи та Учні. Для цього у вікні схеми даних, утримуючи клавішу **Ctrl**, виділіть у таблиці Класи поля паралель та буква і перетягніть їх на таблицю Учні. У вікні **Изменение связей** задайте такі зв'язки, як показано на рис. 11.3. Після цього схема даних має набути такого вигляду, як на рис. 11.2.

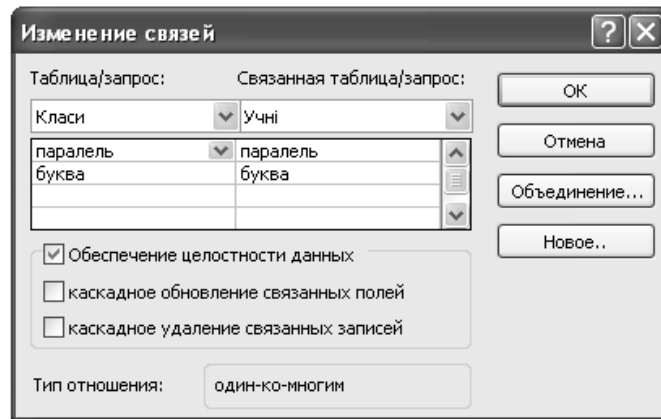


Рис. 11.3. Створення зв'язку зі складеним ключем у вікні **Изменение связей**

6. Створіть зв'язок між таблицями Класи та Викладання.

## Запити на оновлення даних

Повернімося до нашого завдання: автоматично збільшити на 1 значення поля паралель у всіх записах таблиці Учні. Для цього створимо і виконаємо запит на оновлення даних. Як це зробити, описано у вправі 11.3.

**Запит на оновлення даних** призначено для автоматичного змінення значень в усіх або деяких записах певної таблиці. Спосіб змінення визначається окремою формулою для кожного поля, значення якого оновлюється.


### Вправа 11.3

Створіть і виконайте запит *збільшити на 1 значення поля паралель у всіх записах таблиці Учні*.

1. Відкрийте конструктор запитів. У вікні **Добавление таблицы** виберіть таблицю Учні, оскільки запит стосується учнів, після чого клацніть кнопку **Добавить** і закрийте це вікно.
2. У меню **Запрос** виберіть команду **Обновление** або клацніть кнопку **Обновление** на стрічці **Конструктор** — так ви задасте тип запиту. Після виконання цієї команди бланк запиту з'явиться рядок **Обновление**, у якому потрібно ввести формули для обчислення нових значень. Рядка **Вывод на экран** не буде, оскільки запит на оновлення даних ніякої інформації на екрані не відображує.
3. Уведіть у бланк запиту дані за зразком, показаним на рис. 11.4.

Поле:	паралель	
Имя таблицы:	Учні	
Обновление:	"паралель"+1	
Условие отбора:		
или:		

Рис. 11.4. Бланк запиту, що збільшує на 1 значення поля паралель у всіх записах таблиці Учні

4. Запустіть запит на виконання кнопкою  і підтвердьте оновлення записів.
5. Відкрийте таблицю Учні і переконайтеся, що всі значення поля паралель збільшено на 1.

У запиті на оновлення даних не обов'язково оновлювати всі записи якоїсь таблиці. Можна задати умову відбору тих записів, що оновлюватимуться. Наприклад, розглянемо таку ситуацію: із 10Б класу вибуло багато учнів і керівництво школи вирішило об'єднати його з 10А, залишивши для об'єданого класу назву 10А. Тоді в таблиці *Учні* потрібно змінити тільки записи учнів 10Б, надавши полю *буква* цих записів значення А. Бланк запиту *перевести всіх учнів 10Б класу до 10А* виглядає так, як на рис. 11.5.

Поле:	буква	паралель
Имя таблицы:	Учні	Учні
Обновление:	"А"	
Условие отбора:	"Б"	10
или:		

Рис. 11.5. Бланк запиту «перевести всіх учнів 10Б класу до 10А»

Як видно з рис. 11.5, в одному стовпці бланка запиту введено формулу оновлення поля *буква* (формула складається з одного символу «А» — це нове значення поля) та умову відбору записів — рівність поля *буква* символу «Б», а в іншому — тільки умову відбору (рівність поля *паралель* числу 10).

Загальна структура запиту на оновлення даних мовою SQL така:

```
UPDATE ім'я таблиці
SET поле1 = вираз1, поле2 = вираз2, ...
WHERE умова відбору записів
```

Отже, у фразі UPDATE вказують назву таблиці, значення в якій оновлюватимуться, у фразі WHERE — умову, за якою відбираються оновлювані записи, а у фразі SET — список присвоєнь. У лівій частині кожного присвоєння (до символу =) записують ім'я оновлюваного поля, а в правій — арифметичний вираз, за яким обчислюється його нове значення.

SQL-текст запит *перевести всіх учнів 10Б класу до 10А* матиме такий вигляд:

```
UPDATE Учні
SET буква="А"
WHERE паралель=10 AND буква="Б"
```

## Завдання 11.2

Реалізуйте та виконайте запит *перевести всіх учнів 10Б класу до 10А*. Створіть копію файлу бази даних *школа\_кінець\_року*, у якій і виконайте завдання.

## Додавання даних

Коли база даних інтенсивно експлуатується, дані в неї додаються щодня і ви знаєте, як це робити в режимі редагування таблиць або за допомогою форм. Однак зараз нас цікавить той випадок, коли потрібно додати відразу багато записів і робити це вручну занадто довго. Дві найпоширеніші ситуації, коли необхідно додавати великі масиви записів, такі:

- ♦ базу даних реструктуризують, внаслідок чого створюються нові таблиці, у які потрібно додавати дані з наявних таблиць;
- ♦ у базу імпортують дані з зовнішніх джерел (інших баз даних, електронних таблиць тощо).

У першому випадку в MS Access створюють *запит на додавання даних*, а в другому застосовують *майстер імпорту даних*. Розглянемо обидва ці випадки детальніше.

## Додавання даних всередині бази

Повернімося до бази даних *школа*, у якій ще не видаляли відомостей про одинадцятикласників, та застосуємо до випускників інший підхід. А саме, їхні записи ми не видалятимемо, а перенесемо до таблиці *Випускники*. Ця таблиця міститиме всі атрибути таблиці *Учні*, крім атрибута *клас*, який буде замінено атрибутом *ВНЗ* для зазначення, до якого вищого навчального закладу поступив випускник.

Для перенесення даних з однієї таблиці в іншу в межах однієї бази даних потрібно створити два запити: за допомогою запиту на додавання дані буде скопійовано в цільову таблицю, а за допомогою запиту на видалення — видалено з вихідної. Як реалізувати цей підхід на практиці, описано у вправі 11.4.

Запит на додавання даних призначено для копіювання даних з однієї таблиці в іншу. У запиті можна вказати, з яких полів у які копіюватимуться дані, а також якій умові мають відповідати записи, звідки дані копіюватимуться.

## Вправа 11.4


Створіть таблицю *Випускники* та перенесіть до неї відомості про всіх одинадцятикласників.

1. Створіть копію файлу бази даних **школа**, назвавши її **школа\_додані\_дані**. Надалі працюйте з цією копією.
2. Створіть таблицю *Випускники* з атрибутами **код**, **прізвище**, **ім'я**, **дата народження**, **стать**, **успішність** та **ВНЗ**. Для атрибута **код** оберіть числовий тип і задайте розмір поля **Длинное целое**, атрибут **ВНЗ** зробіть текстовим, а всі інші — такими самими, як і в таблиці *Учні*.
3. Створіть запит на додавання відомостей про одинадцятикласників з таблиці *Учні* до таблиці *Випускники*.
  - а) Запустіть конструктор запитів і у вікні **Добавление таблицы** виберіть таблицю *Учні* (ту, з якої додаватимуться записи).
  - б) У меню **Запрос** або на стрічці **Конструктор** виберіть тип запиту — **Добавление**. Буде відображено вікно **Добавление**, у якому зі списку **имя таблицы** потрібно вибрати таблицю *Випускники* (ту, в яку додаватимуться записи) і клацнути кнопку **ОК**.
  - в) Сформууйте бланк запиту такого вигляду, як на рис. 11.6, перетягуючи у його стовпці поля таблиці *Учні*.

Зауважте, що в рядку **Поле** вказано назви полів вихідної таблиці (*Учні*), а в рядку **Добавление** — назви відповідних полів цільової таблиці (*Випускники*). Оскільки поля клас у цільовій таблиці немає, рядок **Добавление** у відповідному стовпці порожній, але натомість слід заповнити рядок **Условие отбора**, адже саме за умовою, накладеною на значення поля клас, відбиратимуться записи учнів для подальшого копіювання.

Поле:	код	прізвище	ім'я	дата народження	стать	успішність	клас
Имя таблицы:	Учні	Учні	Учні	Учні	Учні	Учні	Учні
Сортировка:							
Добавление:	код	прізвище	ім'я	дата народження	стать	успішність	
Условие отбора:							Like "**11"
или:							

Рис. 11.6. Бланк запиту на додавання даних

4. Запустіть запит на виконання кнопкою  і підтвердьте додавання записів.
5. Відкрийте таблицю *Випускники* і переконайтеся, що всі записи одинадцятикласників у неї скопійовано.
6. Видаліть записи про одинадцятикласників з таблиці *Учні*, створивши для цього запит на видалення даних (згадайте вправу 11.1).

Наведемо загальну структуру простого SQL-запиту на додавання даних:

```
INSERT INTO ім'я_цільової_таблиці
           (список полів цільової таблиці)
SELECT список полів вихідної таблиці
FROM ім'я_вихідної_таблиці
WHERE умова відбору записів
```

Зокрема SQL-текст запиту, створеного у вправі 11.4, буде таким:

```

INSERT INTO Випускники (код, прізвище, [ім'я],
                        [дата народження], стать, успішність)
SELECT код, прізвище, [ім'я], [дата народження],
        стать, успішність
FROM Учні
WHERE клас Like "11*"

```

### Завдання 11.3

Створіть таблицю Математики такої самої структури, як і таблиця Учителі, а також запит, що копіюватиме відомості про всіх математиків з таблиці Учителі в таблицю Математики.

### Імпорт даних

Припустимо, створюється нова база даних школи і директор попросив усіх класних керівників надати оператору бази даних списки учнів в електронних таблицях тієї самої структури, що й таблиця Учні. Перед оператором постає завдання імпортувати дані з електронних таблиць у таблицю БД. Його виконують за допомогою майстра імпорту даних, роботу з яким докладно описано у праві 11.5.

### Вправа 11.5

Імпортуйте дані з електронної таблиці в таблицю Учні.

1. Створіть електронну таблицю з відомостями про трьох учнів 9А класу. Щоб не помилитися під час введення даних, експортуйте таблицю Учні у файл MS Excel, а потім змініть у ньому дані за допомогою табличного процесора.
  - а) Відкрийте таблицю Учні і виконайте команду **Файл** ▶ **Експорт** в Access 2003 або клацніть кнопку **Excel** в області **Експорт** на стрічці **Внешние данные** в Access 2007.
  - б) У вікні **Експорт объекта** зі списку формату файлу виберіть **Microsoft Excel 97–2003**, у поле **имя файла** введіть слово **Учні**. Після цього у Access 2003 залишиться клацнути кнопку **Експорт всех**, а в Access 2007/2010 —

кнопку **ОК**, а потім, у вікні **Сохранение шагов экспорта**, — кнопку **Закреть**.

- в) Закрийте таблицю Учні.
- г) Відкрийте електронну книгу **Учні.xls** і залиште в таблиці на аркуші Учні рядок заголовків та три рядки даних. Якщо дані деяких стовпців у клітинки не вміщуються (про це свідчать символи #####), розширте ці стовпці. Змініть код, прізвище, ім'я та дату народження учнів, а також введіть клас — **9А** (рис. 11.7). Стежте за тим, щоб нововведених кодів не було в таблиці Учні, адже значення поля код мають бути унікальними. Збережіть та закрийте електронну книгу.

	A	B	C	D	E	F	G
1	код	прізвище	ім'я	дата народжен	стать	клас	успішність
2	15	Трубач	Василь	21.12.1997	ч	9А	9,9
3	16	Лозова	Ганна	19.09.1997	ж	9А	10,2
4	17	Довгальский	Станіслав	07.10.1997	ч	9А	10,7
5							

Рис. 11.7. Електронна таблиця з відомостями про учнів

2. Імпортуйте відомості з електронної книги в базу даних.

**MS Access 2003**

- а) Виконайте команду **Файл** ▶ **Внешние данные** ▶ **Импорт**, виберіть у вікні **Импорт** тип файлу **Microsoft Excel (\*.xls)**, а потім сам файл **Учні.xls** і клацніть кнопку **Импорт**. Буде запущено майстер імпорту даних.
- б) У перших двох вікнах майстра можна клацнути кнопку **Далее**.
- в) У третьому вікні майстра встановіть перемикач **в существующей таблице** та вкажіть таблицю, де зберігатимуться дані, а в четвертому вікні клацніть кнопку **Готово**.

- а) Клацніть кнопку **Excel** в області **Импорт** на стрічці **Внешние данные**.
- б) У вікні вибору джерела даних за допомогою кнопки **Обзор** виберіть файл електронної книги, установіть перемикач **Добавить копию записей в конец таблицы** та вкажіть у списку праворуч від нього таблицю **Учні**.
- в) Клацніть кнопку **OK**, а потім — кнопку **Готово**.

3. Відкрийте таблицю **Учні** і переконайтеся, що відомості про нових учнів було імпортовано успішно.

## Висновки

- ◆ Запити на додавання, видалення і оновлення даних застосовують тоді, коли потрібно скопіювати, перемістити, додати, видалити або змінити відразу багато записів якоїсь таблиці.
- ◆ Запит на видалення даних призначено для видалення з таблиці записів, що задовольняють певну умову.
- ◆ Запит на оновлення даних призначено для автоматичного змінення значень в усіх або деяких записах певної таблиці. Спосіб змінення визначається окремою формулою для кожного поля, значення якого оновлюється.
- ◆ Запит на додавання даних призначено для копіювання даних з однієї таблиці в іншу. У запиті можна вказати, з яких полів у які копіюватимуться дані, а також якій умові мають відповідати записи, звідки дані копіюватимуться.
- ◆ Щоб перемістити дані з вихідної таблиці в цільову, потрібно спочатку виконати запит на додавання, який скопіює дані у цільову таблицю, а потім — запит на видалення, який видалить їх з вихідної таблиці.
- ◆ Додати дані з зовнішніх джерел, зокрема інших баз даних та електронних книг, можна за допомогою майстра імпорту.

## Завдання для самостійного виконання

1. Реалізуйте і виконайте запити на видалення даних. Для виконання завдань створіть копію бази даних **школа**.
  - а) Видалити відомості про хлопців-десятикласників, чия успішність нижча за 6 балів.
  - б\*)Видалити відомості про те, у яких класах викладає Корбут Василь Петрович.
  - в\*)Видалити відомості про класи, у яких не викладає жодного вчителя-математика.
2. Реалізуйте і виконайте запити на оновлення даних. Для виконання завдань створіть копію бази даних **школа**.
  - а) Перерахувати успішність одинадятикласників з 12-бальної шкали на 100-бальну, помноживши значення поля **успішність** на коефіцієнт  $100/12$ .
  - б\*)Замінити спеціальність усіх фізиків, які викладають у 10 або 11 класах, на «фізика/математика».
  - в\*)Замінити спеціальність класних керівників, які не викладають у 10 та 11 класах, на «класний керівник».

## Питання для роздумів

1. Моделюючи оновлення бази даних після завершення навчального року, ми спочатку видаляли записи одинадятикласників, а потім збільшували на 1 паралель всіх інших учнів. Поясніть, чому не можна виконувати дії у зворотному порядку: спочатку збільшити на 1 паралель всіх учнів, крім одинадятикласників, а потім одинадятикласників видалити?
2. Сформулюйте наведені нижче SQL-запити до бази даних **школа** у словесному вигляді.
  - а) **INSERT INTO** Викладання (учитель, клас)  
**SELECT** паспорт, клас  
**FROM** Учителі, Класи  
**WHERE** Класи.[класний керівник] =  
 Учителі.паспорт

```

б) DELETE *
   FROM Учителі
   WHERE паспорт NOT IN
      (SELECT учитель FROM Викладання)

в) UPDATE Класи
   SET класний керівник =
      (SELECT паспорт FROM Учителі
       WHERE прізвище="Сошко" AND
              ім'я="Катерина")
   WHERE клас = "11Б"

```

## Завдання для досліджень

1. Дізнайтеся, як працює запит на об'єднання таблиць, сформулюйте, реалізуйте та виконайте один такий запит. Для того, щоб створити такий запит, базу даних **школа** слід дещо модифікувати.
2. Дізнайтеся, що таке реплікація бази даних, чим вона відрізняється від резервного копіювання, коли застосовується і як реалізується в MS Access. Створіть репліку бази даних **школа** і продемонструйте роботу з реплікою.

## Додаток

# Опис предметних областей для навчальних проєктів

### Варіант 1. Футбол

Кожний футбольний клуб має назву, рік заснування і розташовується в певному населеному пункті певної країни. У клубу є багато гравців і один головний тренер. Відомі прізвища, імена і дати народження тренерів і гравців. Крім того, про кожного гравця відомо, яке амплуа він виконує на футбольному полі. Клуби з клубами грають матчі, які проходять у певний день на певному стадіоні і завершуються з певним рахунком. Потрібно також зберігати відомості про те, який з клубів був господарем поля, а який — гостем у кожному матчі.

### Варіант 2. Вступ до ВНЗ

Про кожного абітурієнта відомі його прізвище, ім'я, стать і дата народження. Абітурієнт може здавати ЗНО з кількох предметів і отримувати з кожного з них певний бал. Вищі навчальні заклади здійснюють підготовку за кількома напрямками і з кожного з них є перелік предметів для абітурієнтів та прохідних балів ЗНО з кожного такого предмету. Абітурієнт може подавати документи на вступ до кількох ВНЗ, причому на кілька спеціальностей у кожному, але зарахований на навчання може бути тільки в один ВНЗ.

### Варіант 3. Державний устрій

Про держави відомі їхні назви та форми державного устрою (демократія, монархія тощо). Слід зберігати відомості про прізвище, ім'я та посаду голови кожної держави, прізвища, імена та посади членів уряду, а також прізвища та імена всіх громадян. Людина може бути громадянином кількох держав.

### Варіант 4. Мобільний зв'язок

Кожен номер мобільного телефону належить певному оператору, причому за першими трьома цифрами номера оператора

можна визначити однозначно. Потрібно зберігати відомості про те, з яких номерів на які було здійснено телефонні дзвінки, який час початку і тривалість кожного дзвінка. Крім того, кожен номер відповідає певному тарифному плану, що встановлюється оператором. Будемо вважати, що кожен тарифний план, крім назви, характеризується трьома параметрами: вартістю дзвінка, вартістю СМС та вартістю хвилини розмови.

### Варіант 5. Музика

Є відомості про музичні гурти, їх учасників, дискографію та окремі композиції. Щодо кожного гурту відомий рік його заснування та музичний стиль, а щодо кожного учасника — прізвище, ім'я, рік народження та роль у гурті. Гурти випускають альбоми, відомі їхні назви та роки випуску. Альбоми складаються з композицій, що характеризуються назвою і тривалістю. Одна композиція може входити до складу кількох альбомів. У гурту може бути лідер. Музикант може бути лідером тільки одного гурту.

### Варіант 6°. Програмне забезпечення

Щодо кожної програми відома її назва, а також є текст, що описує її призначення. Програма має версії, кожна з яких характеризується номером і датою випуску. Програма може входити до складу програмного пакету і працювати на платформі однієї або кількох операційних систем. Операційні системи мають всі властивості програм і, крім того, характеризуються розрядністю, типом інтерфейсу, можуть бути однозадачними чи багатозадачними.

### Варіант 7°. Водні ресурси

Кожна ріка характеризується назвою, довжиною і площею басейну, а море — площею водної поверхні. Потрібно зберігати відомості про те, територією яких держав протікає ріка, а також у яку водойму (море чи іншу ріку) вона впадає. Штучні моря (водосховища) також характеризуються площею водної поверхні і в них також можуть впадати ріки, однак кожне водосховище, на відміну від природного моря, розташоване на певній ріці.

## Зміст

Розділ 1. Основи баз даних .....	5
Поняття бази даних – Системи керування базами даних – Моделі даних	
Розділ 2. Модель «сутність-зв'язок» .....	16
Об'єкти, сутності, зв'язки – Графічні позначення в моделі «сутність-зв'язок» – Різновиди зв'язків – Ключові атрибути – Побудова моделі «сутність-зв'язок» – Головний принцип семантичного моделювання	
Розділ 3°. Поглиблене семантичне моделювання.....	31
Обов'язковість зв'язків – Слабкі сутності – Зв'язок «є» – Зв'язок між кількома сутностями – Зв'язок сутності самої з собою	
Розділ 4. Операції з таблицями.....	47
Створення бази даних у СКБД MS Access – Основні об'єкти бази даних Microsoft Access – Створення таблиць – Операції з наявними таблицями – Уведення та редагування даних – Збереження та відкриття бази даних	
Розділ 5. Створення зв'язків між таблицями.....	66
Створення зв'язку «один-до-багатьох» – Забезпечення цілісності даних – Створення зв'язку «багато-до-багатьох» – Створення зв'язку «один-до-одного» – Реалізація концепцій поглибленого семантичного моделювання	
Розділ 6. Інтерфейс користувача бази даних .....	92
Створення форм у режимі майстра – Редагування форм у конструкторі	
Розділ 7°. Автоматизація роботи з базою даних.....	106
Інтерфейс для введення інформації про зв'язки – Навігація базою даних	
Розділ 8. Вибір даних.....	119
Вибір даних з однієї таблиці – Вибір даних з кількох таблиць	
Розділ 9°. Основи мови запитів .....	136
Загальна структура SQL-запиту – З'єднання таблиць – Підзапити – Віднімання множин записів	
Розділ 10. Групування даних.....	150
Групові операції в запитах – Групування в мові SQL° – Звіти	
Розділ 11°. Автоматизоване видалення, оновлення і додавання даних .....	172
Видалення даних – Оновлення даних – Додавання даних	
Додаток. Опис предметних областей для навчальних проектів .....	189



Для замовлення книг звертайтеся за  
тел. 050-648-05-00  
ел. пошта [ihorza@gmail.com](mailto:ihorza@gmail.com)  
сайт <http://zavadsky.at.ua>

Навчальне видання

**Ігор Олександрович Завадський**  
**ОСНОВИ БАЗ ДАНИХ**

Комп'ютерна верстка О.М. Левченко  
Коректор В.В. Завадська

ФОП І.О. Завадський  
Свідоцтво про внесення суб'єкта видавничої справи  
до Державного реєстру видавців, виготівників  
і розповсюджувачів видавничої продукції  
серія КІ № 151 від 07.07.2011

Підписано до друку 3.08.11. Формат 60×84/16  
Ум. друк. аркушів 12

Віддруковано на ДП  
«Вінницька картографічна фабрика»  
21100, м. Вінниця, вул. 600-річчя, 19

Свідоцтво про внесення суб'єкта видавничої справи  
до Державного реєстру видавців, виготівників  
і розповсюджувачів видавничої продукції  
ДК № 869 від 26.03.2002 р.