

ЛУЦЬКИЙ ІНСТИТУТ РОЗВИТКУ ЛЮДИНИ
УНІВЕРСИТЕТУ «УКРАЇНА»

кафедра “Комп’ютерні технології”



СУБД Microsoft Access

Частина 1

***Методичний посібник
для самостійного вивчення***

для студентів 1 курсу
денної та заочної форм навчання
напрямків

6.010203 - «Здоров'я людини»

6.030402 - «Правознавство»

5.060101 - «Правознавство»

6.030507 - «Маркетинг»

6.020107 - «Туризм»

ЛУЦЬК 2010

УДК 004(075.8)

ББК 32.97я73

С 34

СУБД Microsoft Access. Методичний посібник для самостійного вивчення. Частина 1. / Укладачі Л.А. Сидорчук., І.М. Поліщук. – Луцьк: ЛІРОЛ Університету "Україна", 2010.– 44с.

Укладачі:

- Сидорчук Л.А., викладач кафедри комп'ютерних технологій ЛІРОЛ Університету "Україна"
- Поліщук І.М., викладач кафедри комп'ютерних технологій ЛІРОЛ Університету "Україна"

Рецензенти:

- А.П. Усійчук, доцент, зав. кафедри комп'ютерної інженерії Луцького інституту розвитку людини Університету "Україна"
- Бурчак І.Н. к.т.н., доцент кафедри інженерної та комп'ютерної графіки ЛНТУ

Схвалено кафедрою комп'ютерних технологій
ЛІРОЛ Університету "Україна" (протокол № 10 від 28 травня
2010р.)

Рекомендовано до друку Вченою радою ЛІРОЛ Університету
"Україна" (протокол № _____ від _____ р.)

©Сидорчук Л.А., Поліщук І.М., (укладання), 2010

Вступ

Під базою даних розуміють сукупність даних, приклади та програми для роботи з даними, інтегрованість даних, тобто їх доступність для багатьох користувачів.

Система управління базами даних — це прикладна програма, реалізована на електронній обчислювальній машині чи обчислювальному комплексі. За допомогою її можна: створювати структуру бази даних, вводити інформацію та зберігати її на зовнішніх носіях; виконувати певне коло операцій з даними; одержувати результати та зберігати їх на зовнішніх носіях або передавати на віддалені термінали; виводити інформацію на термінал у зручній для користувача формі або на друкувальні пристрої; давати можливість працювати з базами даних багатьом користувачам. У цьому визначенні відсутній людський фактор — персонал, який відповідає за дані, адміністратор бази даних, але для розуміння роботи СУБД буде достатньо попереднього визначення.

1. Основні концепції бази даних.

Під час структуризації інформація набуває вигляду даних. Рівень структуризації інформації в певній предметній галузі залежить від рівня вивченості цієї предметної галузі, від рівня розвитку науки, яка займається нею. З вивченням певної предметної галузі відбувається структуризація інформації у вигляді даних. Дані є неподільною часткою інформації, вона несе у собі неподільний згусток інформації і є такою ж абстракцією, як квант енергії в ядерній фізиці. Наприклад: дата народження, ідентифікатор податкоплатника, поштова адреса та ін. Так, як атоми складають молекули, так і дані складають записи. Запис — це є скінченна сукупність даних, який несе певний обсяг інформації про цей об'єкт. Цей обсяг інформації визначається, по-перше, предметною галуззю, в якій розглядається об'єкт, а по-друге, тією задачею, у якій цей об'єкт розглядається. Наприклад, з погляду відділу кадрів заводу, інформація про працівника має такий вигляд: прізвище, ім'я та по батькові; дата народження; місце народження; освіта; спеціальність; трудовий стаж і т.д.

З погляду бухгалтерії заводу інформація про працівника повинна мати такий вигляд: прізвище, ім'я, по батькові; табельний номер; відділ або цех; посада; заробіток або оклад та ін.

Переходячи до визначення поняття бази даних, слід зауважити, що це поняття протягом тривалого часу уточнювалося, і на даний момент його можна навести у такому визначенні. База даних — це множина даних з певною множиною операцій над даними. Якщо позначити множину даних літерою D , а множину операцій — O , то пара $\langle D, O \rangle$ являтиме собою базу даних. Таке визначення, по суті своїй, близьке до визначення алгебраїчних систем: алгебраїчна система — це множина елементів з певною сукупністю операцій на цій множині. Якщо операція одна, то говорять, що мають справу з групами. Якщо операцій дві, то таку алгебраїчну систему називають кільцем і т. д. Оскільки природа елементів у визначенні не обумовлюється, то цим і характеризується універсальність алгебраїчних систем: їх закони чинні на

множинах будь-яких елементів. Отже, вважатимемо, що множина даних D із сукупністю операцій O є базою даних.

У багатьох посібниках поняття *множина даних* і *множина операцій* розділені. Спочатку базу даних характеризують як сукупність, множину даних, а потім наголошують на тому, що з даними можна робити такі й такі операції. Зауважимо, що, як і за аналогією з алгебраїчними системами, фіксація тих чи інших операцій приведе до розгляду баз даних з різними функціональними властивостями. І справді, останнім часом почали розглядати та досліджувати властивості баз даних з певними наборами операцій.

База даних сама по собі нічого не варта. Потрібен засіб, механізм за допомогою якого можна було б виконувати певні операції в базі даних. Спочатку бази даних являли собою картотеки, де структурна одиниця бази даних — запис — повністю розміщувалася в одній картці. Робота з картками мала такі види: вставляння, вилучення, вибір та оновлення. Робота проводилася вручну, була непродуктивною і трудомісткою. Перші спроби механізації цього процесу були зроблені після винайдення перфокарти, розробки правил кодування інформації та механізмів зчитування інформації. Хоча ця технологія обробки інформації протрималася до кінця 60-х років XX століття (На деяких підприємствах можна було зустріти таку технологію й у 80-х роках, наприклад, підсистеми матеріального обліку чи заробітної плати дуже зручно було вести на той час на перфокартах.), але поняття бази даних вперше з'явилося в праці: *Codd E. A Relational Model of Data for Large Shared Data Banks.* — *CACM* 13, № 6 (June 1970). Стаття містить пояснення реляційної моделі, визначення деяких операцій реляційної алгебри, а також обговорення надлишковості та несуперечності баз даних. Лише з появою ЕОМ та динамічних носіїв інформації почали звертати увагу на розробку теоретичних підвалин обробки інформації.

Нині під базою даних розуміють сукупність даних, приклади та програми для роботи з даними, інтегрованість даних, тобто їх доступність для багатьох користувачів.

Переходимо тепер до основного об'єкта нашого вивчення: системи управління базами даних (СУБД).

Система управління базами даних — це прикладна програма, реалізована на електронній обчислювальній машині чи обчислювальному комплексі. За допомогою її можна: 1) створювати структуру бази даних, вводити інформацію та зберігати її на зовнішніх носіях; 2) виконувати певне коло операцій з даними; 3) одержувати результати та зберігати їх на зовнішніх носіях або передавати на віддалені термінали; 4) виводити інформацію на термінал у зручній для користувача формі або на друкувальні пристрої; 5) давати можливість працювати з базами даних багатьом користувачам. У цьому визначенні відсутній людський фактор — персонал, який відповідає за дані, адміністратор бази даних, але для розуміння роботи СУБД буде достатньо попереднього визначення.

Система управління базами даних дає можливість позбутися ряду недоліків, які раніше притаманні були базам даних:

- 1) може бути значно зменшена надмірність інформації через нормалізацію таблиць, у яких зберігаються дані;
- 2) може бути збережена цілісність та достовірність даних. До появи СУБД ця проблема була нерозв'язною;
- 3) може бути збережена безпека (захист) даних;
- 4) може бути досягнута незалежність даних від багатьох користувачів, що і є основною метою створення систем управління базами даних.

2. Проектування бази даних

Перед тим як створювати таблиці, форми та інші об'єкти, потрібно задати структуру бази даних. Добра структура бази даних є основою для створення адекватної вимогам, ефективної бази даних. Сам процес

проектування бази даних являє собою складний процес проектування відображення опису предметної області у схему внутрішньої моделі даних. Перебіг цього процесу є послідовністю більш простих процесів проектування менш складних відображень. Ця послідовність у процесі проектування весь час уточнюється, вдосконалюється таким чином, щоб були визначені об'єкти, їх властивості та зв'язки, які будуть потрібні майбутнім користувачам системи.

Пропонуємо майбутнім користувачам систем управління базами даних два підходи, два варіанти проектування баз даних. Перший варіант широко відомий, бо він запропонований фірмою Microsoft. Другий варіант відображає практичний досвід проектування, і за основу взято варіант, надрукований у «ComputerWorld — Moscow» за 1996 рік.

Варіант 1. Етапи проектування бази даних

Нижче наведені основні етапи проектування бази даних:

1. Визначення мети створення бази даних.
2. Визначення таблиць, що їх повинна містити база даних.
3. Визначення необхідних у таблиці полів.
4. Завдання індивідуального значення кожному полю.
5. Визначення зв'язків між таблицями.
6. Відновлення структури бази даних.
7. Додавання даних і створення запитів, форм, звітів та інших об'єктів бази даних.
8. Використання засобів аналізу в СУБД.

Розглянемо ці етапи дещо детальніше.

1) Визначення мети створення бази даних. На першому етапі проектування бази даних необхідно визначити мету створення бази даних, основні її функції та інформацію, яку вона повинна містити. Тобто потрібно визначити основні теми таблиць бази даних та інформацію, що міститимуть поля таблиць.

База даних має відповідати вимогам тих, хто безпосередньо з нею працюватиме. Для цього потрібно визначити теми, які повинна покривати база даних, звіти, які вона має видавати, проаналізувати форми, що у даний момент використовуються для запису даних, порівняти створювану базу даних із добре спроектованою, подібною їй базою.

2) Визначення таблиць, які повинні містити база даних. Одним із найскладніших етапів у процесі проектування бази даних є розробка таблиць, тому що результати, які повинна видавати база даних (звіти, вихідні форми тощо), не завжди дають повне уявлення про структуру таблиці. У разі проектування таблиць зовсім не обов'язково використовувати СУБД. Спочатку краще розробити структуру на папері. Отже, у разі проектування таблиць слід керуватися такими основними принципами:

- інформація в таблиці не повинна дублюватися. Не повинно бути повторень і між таблицями. Коли певна інформація зберігається лише в одній таблиці, то і змінювати її доведеться лише в одному місці. Це робить роботу ефективнішою, а також виключає можливість розбіжності інформації в різних таблицях. Наприклад, в одній таблиці мають міститися адреси й телефони клієнтів;

- кожна таблиця повинна містити інформацію лише на одну тему. Дані на кожную тему опрацьовуються набагато легше, якщо вони утримуються в незалежних одна від іншої таблицях. Наприклад, адреси та замовлення клієнтів зберігаються в різних таблицях, щоб у разі вилучення замовлення інформація про клієнта залишилася в базі даних.

3) Визначення необхідних у таблиці полів. Кожна таблиця містить інформацію на окрему тему, а кожне поле в таблиці містить окремі дані по темі таблиці. Наприклад, у таблиці з даними про клієнта можуть бути поля з назвою компанії, адресою, містом, країною і номером телефону. Під час розробки полів для кожної таблиці необхідно пам'ятати:

- кожне поле має бути пов'язане з темою таблиці;

- не рекомендується включати до таблиці дані, що є результатом виразу;
- у таблиці має бути вся необхідна інформація;
- інформацію варто розбивати на найменші логічні одиниці (наприклад, поля «Ім'я» і «Прізвище», а не загальне поле «Ім'я»).

4) Задання індивідуального значення кожному полю. З тим, щоб СУБД могла зв'язати дані з різних таблиць, наприклад дані про клієнта і його замовлення, кожна таблиця повинна містити поле чи набір полів, що задаватимуть індивідуальне значення кожного запису в таблиці. Таке поле чи набір полів називають основним ключем.

5) Визначення зв'язків між таблицями. Після розподілу даних по таблицях і визначення ключових полів необхідно вибрати схему для зв'язку даних у різних таблицях. Для цього потрібно визначити зв'язки між таблицями. Бажано вивчати зв'язки між таблицями в уже існуючій базі даних. Для перегляду зв'язків у вибраній базі даних відкриваємо її і вибираємо відповідні команди.

6) Відновлення структури бази даних.

Після проектування таблиць, полів і зв'язків необхідно ще раз переглянути структуру бази даних і виявити можливі недоліки. Бажано це зробити на даному етапі, поки таблиці не заповнені даними. Для перевірки необхідно створити кілька таблиць, визначити зв'язки між ними та ввести кілька записів у кожену таблицю, потім подивитися, чи відповідає база даних поставленим вимогам. Рекомендується також створити чернеткові вихідні форми та звіти й перевірити, чи видають вони необхідну інформацію. Крім того, необхідно виключити з таблиць усі можливі повторення даних.

7) Додавання даних і створення інших об'єктів бази даних. Якщо структури таблиць відповідають поставленим вимогам, то можна вводити всі дані. Потім можна створювати будь-які запити, форми, звіти, макроси та модулі.

8) Використання засобів аналізу в СУБД. Наприклад, у СУБД Microsoft Access є два інструменти для вдосконалення структури баз даних. **Майстер аналізу таблиць** досліджує таблицю, в разі потреби пропонує нову її структуру та зв'язки, а також переробляє її. **Аналізатор швидкодії** досліджує всю базу даних, дає рекомендації з її поліпшення, а також реалізує їх.

Варіант 2. Розробка проекту бази даних

1) Розробка логічної моделі даних. Логічні моделі використовуються розробниками баз даних для формального представлення інформаційних потреб виробництва, економіки, бізнесу тощо. Найрозповсюдженішою формою відображення цієї моделі слугують ER-діаграми (На використанні різновидностей ER-моделі ґрунтується більшість сучасних підходів до проектування реляційних баз даних. Модель була запропонована Ченом (Chen). Моделювання предметної області базується на використанні графічних діаграм, які включають невелике число різнорідних компонентів. У зв'язку з наочністю подання концептуальних схем баз даних ER-моделі набули значного поширення в системах CASE, які підтримують автоматизоване проектування реляційних баз даних.). Основними поняттями ER-моделі є сутність, зв'язок та атрибут. Кожна з частин такої діаграми повідомляє дещо про структуру даних або про те, як ці дані співвідносяться з іншими.

Як правило, розробка логічної моделі являє собою ітераційний процес, що складається з фаз аналізу, проектування та оцінювання. При цьому на кожній ітерації додаються нові правила. Добрі засоби проектування баз даних мають бути гнучкими, а організація роботи з ними — ефективною. ER-діаграми повинні доповнюватися детальнішою інформацією про бізнес, правила та обмеження посилення на цілісність, а також давати змогу керувати наочним поданням деталей моделі.

Під час створення логічної моделі потрібно насамперед провести важливу роботу з замовником. Найбільший обсяг робіт з базами даних

пов'язаний із запитами. Тож потрібно якнай докладніше дізнатися від замовника про можливі запити до бази даних. Досвід проектування свідчить про те, що замовники часто не уявляють, які можливості даватиме їм база даних, до вирішення яких нових задач вони зможуть долучитися. Через це під час проектування потрібно якнайраніше показати замовникам їхні можливі горизонти, щоб так само якнайраніше довелося б вносити зміни до логічної моделі.

2) Підготовка звіту про логічну модель. Для відстежування процесу проектування логічної моделі використовуються звіти. Вони корисні також для узгодження вимог із замовниками. У звітах, як правило, перераховуються сутності, їх атрибути, правила та обмеження, що вміщують до бази даних. Добрі засоби підготовки звітів містять різні види інформації про логічну модель, сприяють гнучкому розміщенню та форматуванню, а також поданню звіту у файл або його експорту в інші додатки. При узгодженні вимог із замовниками варто результат оформляти окремим протоколом.

3) Перетворення логічної моделі у фізичну. У процесі розробки фізичної моделі сутності, атрибути та зв'язки складають фізичну модель, відображаються у таблиці та стовпчиках. До раніш заданих властивостей стовпчиків (типів даних, протяжностей і невизначених значень) додаються нові — первинні та зовнішні ключі, індекси, перевірочні обмеження та правила підтримки посилкової цілісності. Щоб правильно і добре виконати цей етап проектування, засоби моделювання даних повинні працювати з кількома популярними СУБД SQL-типу, графічно відображати фізичні характеристики, дозволяти призначати та модифікувати тригери за замовчування, створювати власні тригери, денормалізувати фізичну модель, не торкаючись при цьому логічної.

4) Підготовка звіту про фізичну модель. Як правило, для того, щоб переглянути якусь таблицю або всі таблиці одночасно, разом з деталями (стовпчики, їх характеристики, індекси, зовнішні ключі та тригери) застосовують звіт про фізичну модель. Добрі засоби підготовки таких звітів

прості в користуванні, мають гнучкий інтерфейс для задання елементів, що включаються до звіту, організації звіту та його формування. Вони повинні надавати детальну інформацію про реалізацію обмежень, правил посилкової цілісності, включаючи призначення та зміст триггерів.

5) Генерація схеми бази даних. Схема описує реалізацію бази даних з урахуванням специфіки конкретної СУБД. Схема може створюватися або мовою визначення даних (файли DDL), або при прямому зверненні до СУБД. Програмні продукти, які добре підтримують генерацію схеми, дають засоби контролю за генеруючими елементами схеми, що дає змогу зробити цей процес ітеративним. Варто шукати інструменти, які підключаються до нашої цільової СУБД і дають можливість переключатися між різними СУБД, мінімізуючи при цьому ручне редагування.

6) Супроводження розроблюваної моделі даних. Більшість баз даних протягом свого життєвого циклу еволюціонує. Для того, щоб спростити цей процес, рекомендується синхронно змінювати модель та базу даних. Варто звертати увагу на засоби синхронізації, утиліти керування версіями та захисту. За допомогою найзручніших у роботі інструментів можна переносити зміни в обидва боки: з моделі в схему, і навпаки. Якщо раніше замовник після здачі СУБД в експлуатацію відмовлявся від супроводження, то тепер, як правило, проектувальники супроводжують експлуатацію СУБД. Це накладає на них додаткову відповідальність за якість проектування, бо всі негаразди доводиться ліквідовувати їм самим.

7) Звернене проектування, що виходить з існуючої бази даних. Відтворення схеми існуючої бази даних служить кільком цілям. Воно дає змогу побудувати модель цієї бази даних, перенести існуючу базу даних з однієї СУБД на іншу, а також досить просто модифікувати схему бази даних, що функціонує. Ключовими параметрами для виконання такого завдання є точність та гнучкість. Ми повинні мати можливість задати елементи схеми, з якими працюватиме програма, й очікується, що внаслідок генерації схеми

бази даних за відновленою моделлю має з'явитися тотожна копія початкової схеми.

Як бачимо, другий варіант окреслює загальніший підхід до проектування баз даних та враховує відносини з замовником проекту. Розглянемо етапи проектування бази даних на основі засобів СУБД MS Access 97, але перед цим дамо характеристики цій реляційній СУБД.

3. Microsoft Access як реляційна СУБД

Microsoft Access — це функціонально повна реляційна СУБД. У ній передбачені всі необхідні засоби для визначення та обробки даних, а також для керування ними при роботі з великими обсягами інформації.

3.1. Визначення даних і їх збереження

Під час роботи з документом або електронною таблицею ми звичайно цілком вільні у визначенні змісту документа або кожної комірки таблиці. На одній сторінці документа можна розмістити кілька абзаців тексту, таблицю, схему чи кілька стовпчиків даних, використовуючи при цьому різноманітні шрифти. У шапці стовпчика електронної таблиці можна як його заголовок ввести певний текст. Також можна в тому самому стовпчику задати різноманітні числові формати, що залежать від призначення рядків. У текстовому редакторі така гнучкість потрібна для того, щоб помістити ту чи іншу інформацію в потрібне місце на сторінці, а в електронній таблиці мати можливість зберегти вихідні дані, робити необхідні обчислення та подавати результати в потрібному вигляді.

Ця гнучкість забезпечує успішний розв'язок невеликих, добре сформульованих задач. Але коли електронна таблиця містить кілька сотень рядків, а документи складаються з багатьох сторінок, то працювати з ними стає досить важко. З ростом обсягу даних можна виявити, що перевищені встановлені електронною таблицею або текстовим редактором обмеження на пам'ять або ж узагалі вичерпані можливості комп'ютерної системи. Якщо розробляється документ або електронна таблиця, які призначені для інших

користувачів, то стає важко проконтролювати введення нових і використання вже наявних даних. Наприклад, коли в електронній таблиці в одній комірці має зберігатися дата, а в іншій — грошове надходження, користувач зовсім випадково може їх переплутати.

Деякі програми для роботи з електронними таблицями дають можливість визначити всередині електронної таблиці ділянку «бази даних», що може допомогти частково розв'язати ці проблеми. Проте, як і раніше, існуватиме обмеження даної програми на обсяг пам'яті і, до того ж, усе одно не одержимо достатнього контролю над уведенням інформації в комірки ділянки «бази даних». Крім того, якщо потрібно буде працювати не лише з цифровою або текстовою інформацією, то може виявитися, що електронна таблиця не може працювати з інформацією, поданою у вигляді малюнка чи звука.

СУБД дозволяє задавати типи даних і засоби їх збереження. Можна також задати критерії (умови), які СУБД надалі використовуватиме для забезпечення правильного введення даних. У найпростішому випадку умова на значення (validation rule) повинна гарантувати, що не буде введено випадково в числове поле текст. Інші умови можуть визначати ділянку чи діапазони припустимих значень даних. У найдосконаліших системах можна задати відношення між сукупностями даних, які називаються таблицями або файлами, і покласти на СУБД забезпечення сумісності чи цілісності даних. Наприклад, можна змусити систему автоматично перевіряти відношення введених замовлень до конкретних клієнтів.

Microsoft Access надає максимальну свободу в завданні типу даних (текст, числові дані, дати, час, грошові значення, малюнки, звук, документи, електронні таблиці). Можна задати також формати збереження (довжина рядка, точність подання чисел і дати/часу) і подання цих даних у разі виведення на екран або друк. Для впевненості, що в базі даних зберігаються лише коректні значення, можна задати умови на значення різноманітного ступеня складності.

Оскільки Microsoft Access є сучасним додатком Windows, можна використовувати всі можливості DDE (Dynamic Data Exchange, динамічний обмін даними) та OLE (Object Linking and Embedding, зв'язок та впровадження об'єктів). DDE дає змогу здійснювати обмін даними між Access і будь-яким іншим додатком Windows, який підтримує DDE. В Access можна за допомогою макросів або Access Basic здійснювати динамічний обмін даними з іншими додатками. OLE є витонченішим засобом Windows, що дає можливість встановлювати зв'язок з об'єктами іншого додатка чи впроваджувати якісь об'єкти в базу даних Access. Такими об'єктами можуть бути картинки, діаграми, електронні таблиці чи документи з інших додатків Windows, що підтримують OLE.

Microsoft Access може працювати з великим числом найрізноманітніших форматів даних, включаючи файлові структури інших СУБД. Можна здійснювати імпорт і експорт даних із файлів текстових редакторів або електронних таблиць. За допомогою Access можна безпосередньо обробляти файли Paradox, dBASE III, dBASE IV, FoxPro та ін. Можна також імпортувати дані з цих файлів у таблицю Access. На додаток до цього Microsoft Access може працювати з найпопулярнішими базами даних, які підтримують стандарт ODBC (Open Database Connectivity — Відкритий доступ до даних).

3.2. Обробка даних

Робота з даними в текстовому редакторі або електронній таблиці значно відрізняється від роботи з даними в СУБД. У документ, підготовлений за допомогою текстового процесора, можна включати табличні дані та використовувати для їх обробки обмежений набір функцій. Можна виконувати пошук рядка символів у вихідному документі, за допомогою OLE включати в нього таблиці, діаграми чи картинки з інших додатків. В електронній таблиці деякі комірки містять формули, які забезпечують потрібні обчислення або перетворення, а дані, що є для них вхідною інформацією, можна вводити в інші комірки. Дані з електронної таблиці, створеної з якоюсь конкретною метою, дуже важко потім використовувати

для виконання інших завдань. Щоб розв'язати нову задачу, потрібно організувати зв'язок із даними іншої електронної таблиці чи використати обмежені можливості пошуку для копіювання обраної підмножини. СУБД дає змогу працювати з даними, застосовуючи різноманітні засоби. Наприклад, можна виконати пошук інформації в окремій таблиці або створити запит зі складним пошуком по кількох пов'язаних між собою таблицях або файлах. За допомогою однієї єдиної команди можна оновити зміст окремого поля чи кількох записів. Для читання та коригування даних можна створити процедури, що використовують функції СУБД. У багатьох системах є розвинуті можливості для введення даних і генерації звітів.

У Microsoft Access для обробки даних таблиць використовується потужна мова SQL (Structured Query Language — структурована мова запитів). Використовуючи SQL, можна виділити з однієї чи кількох таблиць потрібну для розв'язання конкретної задачі інформацію. Access значно спрощує процес обробки даних. Щоб змусити Microsoft Access виконувати наші завдання, не обов'язково потрібно знати мову SQL.

У Microsoft Access є простий і в той же час багатий на можливості засіб графічної постановки запиту — так званий «запит за зразком» (*QBE, query by example*), який використовується для задання даних, необхідних для розв'язання певної задачі. Використовуючи для виділення та переміщення елементів на екрані стандартні прийоми роботи з мишею в Windows і кілька клавіш на клавіатурі, можна побудувати досить складний запит.

Електронні таблиці і текстові документи є прекрасними засобами для розв'язування «однокористувальницьких» задач, але вони погано пристосовані для роботи в режимі колективного користування. Електронні таблиці також корисні як шаблони для простих форм введення інформації. Електронна таблиця добра як шаблон для рахунка-фактури в невеликій фірмі. Але якщо з розширенням бізнесу починає зростати число співробітників, що вводять у комп'ютер замовлення, то без бази даних не обійтися. Так само електронна таблиця може використовуватися на великих підприємствах для

підготовки співробітниками звітів про свої витрати, але для впорядкування загальної бухгалтерської звітності ці відомості однаково повинні збиратися в базі даних.

У тих випадках, коли виникає потреба в колективному користуванні інформацією, система управління базами даних дає змогу захищати інформацію від несанкціонованого доступу так, що право знайомитися з даними чи коригувати їх одержують лише певні користувачі. Призначена для колективного користування СУБД має засоби, які не дозволяють кільком користувачам одночасно коригувати ті самі дані. Найдосконаліші системи дають змогу, крім того, так групувати внесені зміни (послідовності таких змін іноді називають *транzakціями*), що або жодне з них не буде збережено, або будуть збережені всі зміни. Наприклад, вводячи нове замовлення, потрібно спочатку переконатися в тому, що всі його пункти внесені правильно, а якщо трапилася помилка, то простежити, щоб це замовлення (зміни в базі даних) не записувалося в пам'ять. Окрім того, виникає бажання, щоб ніхто інший не бачив цього замовлення чи його частини доти, доки воно цілком не буде введене в базу даних.

Microsoft Access спроектований так, що він може бути використаний як у ролі самостійної СУБД на окремій робочій станції, так і в мережі — у режимі «клієнт — сервер». Оскільки в Access до даних можуть мати доступ відразу кілька користувачів, у ньому передбачені надійні засоби захисту та забезпечення цілісності даних. Можна заздалегідь визначити, які користувачі або групи користувачів матимуть доступ до об'єктів (таблиць, форм, запитів) бази даних. Access також розпізнає та враховує захисні засоби інших, приєднаних до бази даних структур.

Microsoft Access, володіючи всіма рисами класичної СУБД, дає й додаткові можливості. Access — це не лише потужна, гнучка та проста у використанні СУБД, а й система для розробки додатків, що працюють із базами даних. Використовуючи запити, можна вибирати і обробляти інформацію, що зберігається в таблицях. Можна створювати форми для

введення, перегляду та відновлення даних, а також використовувати Access для складання як простих, так і складних звітів. До числа найпотужніших засобів Access відносяться засоби розробки об'єктів — **Майстри**, що їх можна використовувати для створення таблиць, запитів, різноманітних типів форм і звітів.

Access переважно застосовується в чотирьох окремих сферах: на малому підприємстві, у роботі за контрактом, у великих корпораціях, а також у сфері домашнього застосування.

4. Архітектура Microsoft Access

У Microsoft Access *об'єктами* називається усе, що може мати ім'я (у розумінні Access). У базі даних Access основними об'єктами є таблиці, запити, форми, звіти, макроси та модулі.

Таблиця — це об'єкт, який визначається і використовується для збереження даних. Кожна таблиця включає інформацію про об'єкт певного типу, наприклад про клієнтів. Таблиця містить *поля (стовпчики)*, у яких зберігаються різноманітні дані, наприклад прізвище чи адреса клієнта, і *записи* (які називаються також *рядками*). У записах зібрана вся інформація про конкретний предмет (людину, зразок продукції). Для кожної таблиці можна визначити *первинний ключ* (одне чи кілька полів, що мають унікальне значення для кожного запису) та один або кілька *індексів* з метою збільшення швидкості доступу до даного.

Запит — це об'єкт, який дає змогу користувачеві одержувати потрібні дані з однієї чи кількох таблиць. Для створення запиту можна використовувати QBE (запит за зразком) або інструкції SQL. Можна створювати запити на вибір, відновлення, вилучення чи додавання даних. За допомогою запитів можна створювати нові таблиці, використовуючи дані однієї чи кількох таблиць, що вже існують.

Форма є об'єктом, який призначений переважно для введення даних, відображення їх на екрані чи керування роботою додатка. Можна

використовувати форми для того, щоб реалізувати вимоги користувача до введення даних із запитів або таблиць. Форми можна також роздруковувати. За допомогою форми ви можете у відповідь на якусь подію запустити **макрос** або **процедуру** — наприклад, запустити макрос, коли змінюється значення певних даних.

Звіт є об'єктом, що призначений для створення документа, який згодом може бути надрукованим або включеним у документ іншого додатка. Перед тим, як виводити звіт на принтер, можна попередньо переглянути його на екрані.

Макрос є об'єктом, що являє собою структурований опис однієї чи кількох дій, які повинен виконати Access у відповідь на певну подію. Наприклад, можна визначити макрос, який у відповідь на вибір якогось елемента в основній формі відкриває іншу форму. За допомогою іншого макроса можна здійснювати перевірку значення певного поля при зміні його змісту. У макрос можна включати додаткові умови для виконання або невиконання тих чи інших включених у нього дій. Можна також, перебуваючи в одному макросі, запустити інший макрос або функцію модуля.

Модуль є об'єктом, що містить програми на Visual Basic і дає змогу розбити процес на дрібніші дії та виявити ті помилки, які неможливо було знайти з використанням макросів. Модулі можуть бути незалежними об'єктами, що містять функції, які можна викликати з будь-якого місця додатка, але вони можуть бути і безпосередньо прив'язані до окремих форм або звітів для реакції на ті чи інші зміни, що відбуваються в них.

5. Таблиці і поля

Основний структурний елемент Access — *таблиця*, в якій зберігається інформація. Об'єкт «таблиця» — це лише одна частина Access-системи, в якій справді зберігається інформація. Всі інші об'єкти (такі, як запити, форми та звіти) ґрунтуються на даних таблиць.

Для більшості користувачів операції, що виконуються в базі даних, починаються зі створення однієї чи більше таблиць. І хоча, з одного боку, *таблиця* — це колекція даних, з іншого — це дещо більше, ніж просто набір даних. Яка ж відмінність таблиці від сторінки тексту або чисел у структурі електронної таблиці? Структура перетворює дані в інформацію. Структуровану інформацію, організовану в таблицю, легше зрозуміти і читати.

По-перше, за рахунок класифікації по стовпчиках. Кожен стовпчик таблиці являє собою результат певної класифікації.

По-друге, тому, що рядки повторюють шаблон. Шаблон, установлений за допомогою стовпчиків, повторюється в кожному рядку. Кожний рядок подає інформацію про певний існуючий об'єкт, наприклад, про легковий автомобіль, заводи-виготовлювачі або роки випуску.

Ці два чинники дозволяють легко читати інформацію таблиці, оскільки кожен рядок у таблиці передбачений.

Ця передбачена структура дає можливість комп'ютерній програмі виконати аналогічні операції набагато швидше і з вищою точністю, ніж це може зробити людина.

5.1. Стовпчики і рядки, поля та записи

Усі бази даних мають двовимірну структуру. Якщо структуру розуміти як таблицю, то, природно, використовуються терміни *стовпчик* і *рядок*. Загалом, поле є синонімом *стовпчика*, а запис — синонімом *рядка*.

Терміни *рядок* і *стовпчик* застосовуються, коли йдеться про фізичну структуру таблиці, що містить інформацію. *Поле* та *запис* слугують для вираження логічного зв'язку елементів даних, оскільки поля й записи не завжди постають у формі рядків і стовпчиків. Наприклад, усі поля форми належать одному запису, навіть коли у формі відсутні фізично рядок і стовпчик.

Поля визначають класифікаційну характеристику даних, за якою можна знайти кожен запис, такий, наприклад, як прізвище чи дата народження. Коли

працюємо з полем, то можна маніпулювати даними, що належать одному чи кільком записам. Наприклад, за потреби відсортувати інформацію вибираємо одне або більше полів у ролі ключів сортування.

Запис містить інформацію про окрему особу, місце чи предмет.

Терміни *рядок*, *стовпчик*, *поле* та *запис* використовуються в різних частинах системи Access.

5.2. Робота з вікном бази даних

Перш ніж перейти до створення таблиць, потрібно створити базу даних, у якій зберігатимуться таблиця та інші об'єкти Access. Із запуском Access з'являється діалогове вікно (рис. 1).

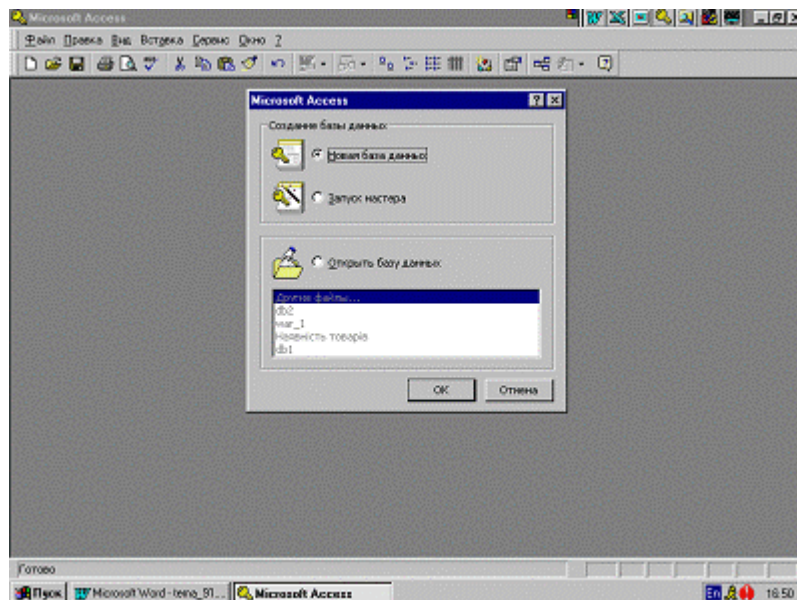


Рис. 1

Можна відкрити вже існуючу базу даних, створити нову базу даних або запустити **Мастера** для створення нової бази даних за зразком. Після відкриття наявної бази даних з'явиться вікно з такими вкладками: **Таблицы**, **Запросы**, **Формы**, **Отчеты**, **Макросы**, **Модули**. На кожній вкладці є кнопки: **Запуск**, **Конструктор**, **Создать**. Натиснувши на одну з цих кнопок, починаємо певну роботу з базою даних. Роботу можна також почати, натиснувши на об'єкті правою кнопкою миші. З'являється відповідне контекстне меню, в якому вибираємо відповідний пункт і починаємо роботу.

5.3. Створення нової таблиці

Кожна база даних може містити одну чи більше таблиць інформації. Нові таблиці створюються натисканням кнопки **Создать** у вікні бази даних, якщо обраний корінець **Таблицы**. У результаті цього відкривається вікно **Новая таблица**, показане на рис. 2.

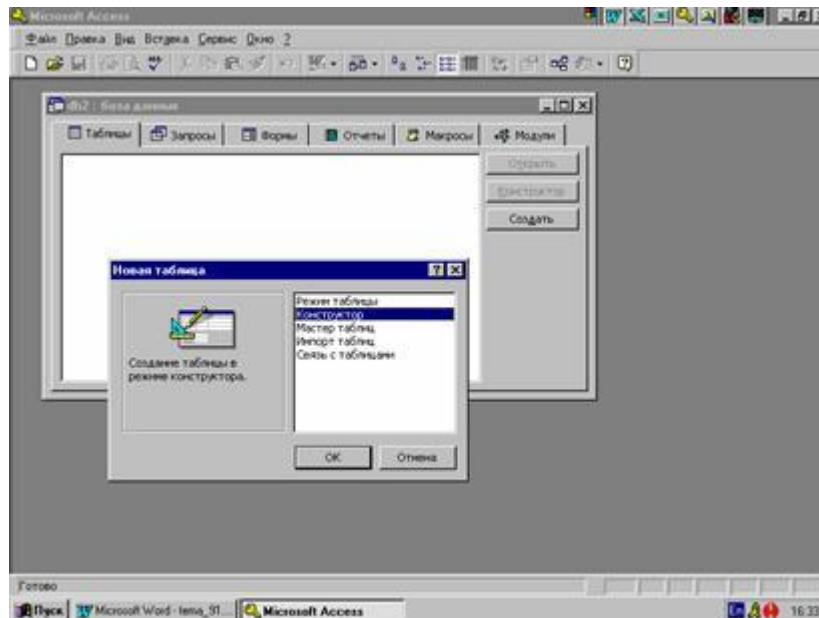


Рис. 2

У правій частині вікна наведено п'ять пунктів, які дають можливість вибрати спосіб задання таблиці **Режим таблицы**, **Конструктор**, **Мастер таблиц**, **Импорт таблиц** і **Связь с таблицами**. Вибираємо пункт **Конструктор** і натискаємо **ОК**.

Вікно конструктора таблиці має три стовпчики: **Имя поля**, **Тип данных** і **Описание** (рис. 3).

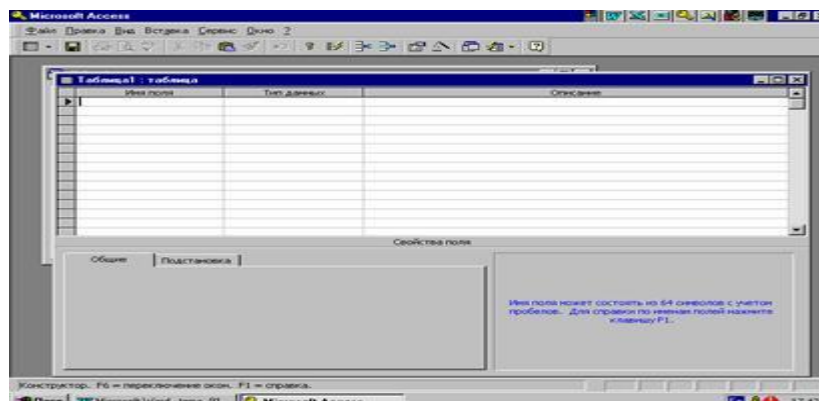


Рис. 3

Ім'я поля — це ім'я, що присвоюється даному полю. Кожне поле в одній таблиці мусить мати унікальне ім'я, а також може мати протяжність не більше 64 символів і містити літери, цифри, проміжки та знаки пунктуації. Бажано у разі надання імен полям дотримуватися таких правил. Імена повинні бути набрані латинськими літерами і завдовжки не більше 8 символів, тобто слід намагатися дотримуватися правил іменування об'єктів у MS DOS. Після введення імені поля натискуємо на клавішу <TAB>, щоб перевести курсор у стовпчик **Тип данных**. За замовчування дане вважається текстовим, завдовжки 50 символів. Щоб змінити тип даних, натискуємо на кнопку розкриття списку та вибираємо потрібний тип. У Access є дев'ять типів даних і майстер підстановок: **Текстовый, Поле МЕМО, Числовой, Дата/время, Денежный, Счетчик, Логический, Поле объекта OLE, Поле типа Гиперссылка, Мастер подстановок**. Вибравши потрібний тип поля, знову натискуємо клавішу Tab, і курсор переходить у поле **Описание**. **Описание** — це записаний нами коментар, у якому мовиться про призначення даного поля. Ці нотатки заносити не обов'язково, але якщо проект виконується для замовника, то опис повинен бути зроблений, щоб не мати клопотів під час здавання проекту.

Говорячи про імена полів, слід зупинитися на одному моменті. Як уже зазначалося вище, бажано давати імена полям за правилами DOS. Але коли проект досить великий і даних багато та ще виникає бажання давати імена більш інформативні, то варто скористатися одним прийомом. Ім'я може складатися з кількох слів, початкові літери цих слів великі, пробіли між словами відсутні. Наприклад: ПосівнаПлоща, ТабНомер, ПосадОклад, ПреміяКвартальна, ПреміяЗаВислугу і т. ін. Access дає можливість використовувати пробіли в іменах полів, тому можна вводити фрази, подібні таким: «Премія за вислугу», але важко розібратися, скільки прогалин між словами у чужих імен полів. Користувачі, які працюватимуть з нашими додатками, теж не зможуть відразу розібратися з іменем поля (Для вказівки поля у формах і звітах Access дає змогу автоматично використовувати для

кожного поля властивість Подпись поля (Caption), замість імені поля. Це означає, що можна використовувати скорочення, подібні до ПП, ПЗВ, у ролі імені поля при складанні виразів і формул для спрощення їх написання та використовувати речення, подібні до Посівна Площа, Премія За Вислугу, для створення зрозумілішого імені поля за появи його лише у формах або звітах.).

5.4. Типи полів

Поле може належати до одного з десяти типів, описаних нижче.

Текстовый (Text) — використовується для збереження будь-якої послідовності символів, включаючи літери, цифри, знаки пунктуації, прогалини та спеціальні символи. Текстові поля можуть бути завдовжки до 255 символів. За замовчування довжина текстового поля дорівнює 50 символам. Найчастіше цей тип використовується в полях, призначених для збереження коротких текстових характеристик, наприклад, імен або адрес.

МЕМО (Memo) — поля цього типу використовуються для збереження довгих текстових блоків до 65 000 символів (близько 16 сторінок інформації). Потрібно використовувати цей тип поля тоді, коли треба ввести чи імпортувати великий текстовий блок. Це поле змінної величини, і задавати розмір його не потрібно.

Числовой (Number) — використовується для збереження числових значень (крім грошових сум). У числових полях, як правило, містяться значення, над якими виконуватимуться обчислення; числа можуть містити десяткові коми та знак «мінус». Якщо число є іменем якогось об'єкта (поштовий індекс, шифр виробу), то потрібно задавати тип поля як **Текстовый**.

Дата/Время (Date/Time) — використовується для зберігання дати/часу, значення вводяться в поля такого типу в стандартному форматі, наприклад 1/1/99/, 12:53:00 PM.

Денежный (Currence) — використовується для збереження числових значень. Автоматично додає фіксовану кількість цифр після десяткової коми, щоб уникнути помилок округлення.

Счетчик (AutoNumber) — автоматично присвоює унікальне числове значення для кожного запису, починаючи з 1 для першого запису, 2 для другого і т. д. Якщо запису присвоєно таке значення, то воно далі не буде змінюватися чи використовуватися знову в тій самій таблиці. Ми не зможемо змінити значення поля такого типу.

Логический (Yes/No) — цей тип поля використовується для збереження логічних величин, які можуть набувати лише двох значень типу «так» чи «ні».

OLE Object — цей тип поля містить дані з інших додатків, які підтримують технологію Object Linking and Embedding (OLE) — зв'язок та впровадження об'єктів. Цей тип поля дає змогу зберегти документи, таблиці, графіки, малюнки, звукові сигнали, відео- та іншу інформацію, створену додатками Windows, що підтримують технологію OLE.

Гиперссылка (Hiperlink) — поле цього типу містить адресу гіперпосилання, яка вказує шлях до іншого об'єкта, документа чи Web-сторінки. Наприклад, якщо таблиця створюється для відстежування документів, створених у Microsoft Word, варто ввімкнути поле гіперпосилання, натиснувши на яке зможемо побачити зміст цих документів.

Мастер подстановок (Lookup Wizard) — поле цього типу використовується для задання списку значень, які може приймати дане поле.

5.5. Використання засобу «Описание»

Після вибору потрібного типу поля натискаємо клавішу Tab і переміщуємося у стовпчик «Описание». Сюди можна за потреби занести довідкову інформацію. Після введення опису чи в разі пропуску натискаємо клавішу Tab. Вказівник зміститься на наступний рядок. За потреби продовжуємо вводити імена полів, типи даних та опис для решти полів.

5.6. Властивості поля

На додачу до трьох основних елементів (**Тип поля**, **Тип данных**, **Описание**) поле може містити також і набір властивостей. *Властивість* — це така характеристика об'єкта, як розмір, тип або ім'я. Кожен об'єкт у Access

має власний набір властивостей. Access має дванадцять властивостей, що можуть бути притаманні полю. Властивості поля залежать від його типу. Наприклад, єдина властивість «Подпись поля» (Caption) притаманна всім полям. Варто також пам'ятати, що вибір властивостей істотно залежить від типу поля. Наприклад, поля типу «Текстовый» (Text) і «Числовой» (Number) мають властивість «Формат поля» (Format). Проте вона не однакова для цих двох типів полів, оскільки логічно вони обробляються по-різному.

За винятком властивостей «Размер поля» (Field Size) і «Формат» (Format), властивості діють на всі типи полів однаково.

«Размер поля» (Field Size). Розмір поля можна зазначити для полів типу «Текстовый» (Text) і «Числовой» (Number). Розмір текстового поля можна легко уявити. Просто вводимо значення, яке встановлює максимальне число символів поля не більше 255. За замовчуванням встановлюється розмір 50. Розмір поля можна не змінювати, бо на відміну від dBASE, Paradox і FoxPro, Access не використовує поля фіксованої довжини, тому на диску зберігаються лише введені в поля символи. За замовчуванням розмір числового поля встановлюється зі значенням Double (подвійної точності), і можна лишити його без зміни.

5.7. Формати полів (Format)

За винятком полів типу OLE Access по-різному здійснює введення та виведення даних. Наприклад, якщо ви в поле вводите число 71500.0, то може знадобитися вивести його у вигляді \$71,500.00. Це можна реалізувати через вибір формату поля, що «змусить» виводити всі введені дані саме в такому вигляді. Форматовані дані, як правило, легше читати, особливо якщо вони числові, тому що послідовність таких символів, як коми, знаки долара та десяткові крапки, не залежить від способу введення даних. Форматування також усуває необхідність введення незначних символів — таких, як знак долара чи кома, що можна вводити автоматично, залежно від формату.

Access забезпечує список попередньо визначених форматів для полів типу **Число**, **Валюта** і **Дата**. Крім того, Access розпізнає спеціальні коди, що задаються форматом, розробленим користувачем.

5.8. Маска введення (Input Mask)

Маски введення використовуються, коли дані, що вводяться в поля, мають наперед визначену структуру. Прикладами можуть бути такі дані, як дата, номери телефонів і факсів, табельний номер тощо. Маска введення виконує дві функції. По-перше, обмежує введення за рахунок установленної фіксованої довжини і типу символу. По-друге, може автоматично під час введення вставляти фіксовані символи. У цьому відношенні добрим прикладом є номер телефону. Маска введення складається з буквених символів — дефісів і дужок — і символу #, замість якого повинна ставитися цифра чи пробіл: (###)-###-##-##. Якщо вводимо число 0444412070, то воно з'явиться в полі у вигляді (044)-441-20-70. Існують різноманітні засоби керування запровадженням даних за допомогою масок. За детальнішою інформацією відсилаємо до [...].

5.9. Підпис поля (Caption)

Дана властивість виражається ім'ям, фразою чи описом, що використовуються за замовчування в ролі заголовка стовпчика або мітки поля. Ця властивість дає змогу використовувати прості імена полів замість занадто інформативних фраз типу *Дата дня народження*, що з'являються в таблицях і формах. Прості імена полів легше використовувати для написання виразів, макросів і програм на VBA.

5.10. Значення за замовчування (Default Value)

Ця властивість використовується, коли додається новий запис для автоматичного введення значень у поле. Звичайно такими значеннями є літери — такі, як О, для поля «Кількість». Можна також використати системну функцію, таку, як Date(), для введення за замовчуванням поточної дати. Access уставляє значення за замовчуванням тільки один раз для кожного нового запису, коли він уперше додається до таблиці. За вилучення вмісту

поля Access лишає поле порожнім. Знову вставлятися значення за замовчуванням не буде.

5.11. Умова на значення (Validation Rule) і повідомлення про помилку (Validation Text)

Подібно до правил введення, умова на значення використовується для контролю введення в поле інформації. Відмінність полягає в тому, що маски введення управляють введенням, базуючись на структурі (такій, як кількість і тип символів, що вводяться), а умови на значення оцінюють зміст вводимого поля. Наприклад, можна мати таку умову на значення, що перевірятиме наявність цифри замість знака # між числами 0 і 100. Правила для перевірки є виразами, що використовуються Access для визначення того, чи буде вводимий елемент збережений як зміст поля. Такі вирази обмежать дати, що можуть бути введені в поле за 15 днів до або після поточної системної дати:

Between Date()-15 And Date()+15

Повідомлення про помилки — це повідомлення, виведене в діалоговому вікні за помилкового введення даних користувачем. Якщо не ввести перевірочний текст, то у разі введення помилкових даних Access виведе в діалоговому вікні таке повідомлення: «The value you entered is prohibited by the validation rule set this field» (Введене вами значення не може бути прийняте на збереження в базу даних відповідно до встановленого правила для даного поля).

Примітка. Здійснюючи введення, керуючись установленим правилом, можна затриматися під час введення вмісту чергового поля, тому що Access не дозволить переміститися до іншого поля, закрити таблицю чи змінити режим роботи доти, доки введені дані будуть помилковими. Потрібно натиснути клавішу Esc для відмови від введення і повернутися до останнього збереженого значення даного поля. Це допоможе вийти з халепи, в яку потрапили через уведення помилкових даних, що контролюються умовою на

призначення. У разі повторного натискання клавіші Esc ми відмовляємося від усіх змін, зроблених у поточному записі.

5.12. Обов'язкове поле (Required)

Ця властивість «змушує» користувача ввести дані в поле. Access не збереже запис доти, доки буде нульове значення хоча б одного поля, для якого встановлене значення Yes (Так) властивості Required. Нульове значення — це незаповнене поле. У числовому полі нуль не є таким і розглядається як припустиме введення. У текстовому полі Access розглядає прогалину як нульове значення. Ми повинні ввести принаймні один видимий символ у поле зі значенням (Так) властивості Required (Обов'язкове поле).

Потрібно пам'ятати, що Access не примушує вводити дані в поля такого типу доти, доки ми не спробуємо зберегти запис. Тільки за такої спроби виводиться вікно з повідомленням для кожного незаповненого поля. Ці вікна з повідомленнями з'являються в тому ж порядку, в якому зазначені поля з'являються в структурі таблиці. Якщо ми вимагаємо обов'язкового заповнення полів *Рік народження* й *Місце народження*, Access виведе вікно, в якому повідомлятиметься про те, що не заповнене поле *Рік народження*. Лише після його заповнення у вікні з'явиться наступне повідомлення про незаповнене поле *Місце народження*.

5.13. Порожні рядки (Allow Zero Lenght)

Ця характеристика допоможе встановити різницю між полями, які були визначені, як поля, що не мають певного значення (для цього ставимо " " в поле), і як поле, що не було заповнене. Нехай, у нас є поле для введення номера посвідчення водія. Якщо в ньому немає інформації, то це може бути витлумачено подвійно. **Порожньо:** ми знаємо, що немає ніякого номера для цього запису. **Невідомо або не існує:** для поля типу *не існує* такий номер може існувати чи не існувати, тож під час введення запису це поле було поки що пропущено. У такому разі іноді можна ввести слово *невідомо*.

У деяких випадках може бути корисно або потрібно розуміти різницю між полями типу *порожньо* (коли немає інформації) і *невідомо* (коли дані

можуть бути введені пізніше). Звичайно, якщо властивість Allow Zero Length має значення **No** (Немає), будь-яким записам, що залишилися незаповненими, автоматично присвоюється нульове значення. Поля обробляються при цьому як порожні. Якщо ми хочемо зазначити, що окремі поля повинні мати значення *невідомо*, то в поле необхідно ввести слова Unknown або None. За такого підходу складність полягає в тому, що користувачі мають завжди правильно вводити слова: Unknown, None і т. ін. Якщо для властивості Allow Zero Length установлене значення Yes, ми можемо вказати на порожнє поле введенням двох лапок (" ") і вказати на поле типу *невідоме*, залишивши його порожнім. Можна використовувати цей підхід для будь-якого текстового поля, не присвоюючи якоїсь спеціальної фрази порожнім записам.

Пізніше ми зможемо використати оператор Is Null для вибору невідомих записів, а для вибору записів, що містять порожнє значення, як умову вибору слід буде використати (" ").

5.14. Індексоване поле

Якщо як значення властивості вибрано **Yes**, Access створює і підтримує індекс значення, введеного в поле Indexed. Індекс — це відсортований список змісту поля з покажчиком, за яким можна знайти місце розташування кожного значення в таблиці. Використання індексу набагато прискорює процес пошуку інформації з певної теми.

Access підтримує два типи індексів.

Індекс — первинний ключ (Primary key). Це поле або комбінація полів, що однозначно визначає кожний запис. Для кожної таблиці має бути визначений первинний ключ. Під час вибору первинного ключа Access генерує індекс — *первинний ключ*, що використовується для визначення порядку виведення записів. Якщо для таблиці первинний ключ не визначений, то записи з'являтимуться в тому порядку, в якому були введені. Індекс — первинний ключ не може мати однакових чи нульових значень. Якщо ми вибрали поле або поля в ролі первинного ключа, які в даний момент

мають однакові чи нульові значення, то первинний ключ не буде встановлений. Хоча Access не потребує встановлення первинного ключа, все одно рекомендується це зробити під час пошуку даних і в реляційних базах даних для встановлення зв'язків між таблицями.

Індекс — поле. Індексне поле не використовується для визначення порядку проходження записів у таблиці. Такі індекси починають працювати, коли Access потрібно відсортувати чи відшукати записи, використовуючи дані у певних полях.

5.15. Установка індексів — первинних ключів

Таблиця може мати лише один індекс — *первинний ключ*. Значення первинного ключа визначають порядок, у якому виводитимуться записи.

У режимі конструктора таблиці можна задати первинний ключ двома способами.

За допомогою піктограми: виберемо поле або поля, які ми хотіли б включити в *первинний ключ*. Клацаємо на піктограмі «ключ» основної панелі інструментів Access. Access видає підтвердження, що ці поля є частиною первинного ключа, через виведення на екран ключа в лівій частині поля первинного ключа.

За допомогою діалогового вікна: діалогове вікно для установки індексів буде виведено на екран за директивою **Індекси** меню **Вид**. Це діалогове вікно містить таблицю з трьома стовпчиками: **Індекс**, **Ім'я поля**, **Порядок сортировки**. Вводимо ім'я, поле і порядок для першого поля, який ми хочемо використовувати як первинний ключ. Access розташовує піктограму «ключ» перед полем «Індекс».

Як би не був установлений первинний ключ, властивості його індексу можна змінювати за допомогою діалогового вікна індексу. Наприклад, можна використовувати це діалогове вікно для зміни порядку сортування за первинним ключем (за зростанням або за спаданням).

Для детальнішого ознайомлення з індексами та роботою з ними можна звернутися до пункту меню **Справка**.

Питання для самоперевірки

1. Що таке бази даних?
2. Що таке реляційна база даних?
3. Назвіть основні елементи вікна СУБД Access?
4. Що таке таблиця?
5. Що таке форма?
6. Що таке звіт?
7. Що таке запит?
8. Що таке макроси?
9. Що таке модулі?
10. Яким чином створюються таблиці в БД?
11. Яким чином створюються форми в БД?
12. Яким чином створюються запити в БД?
13. Яким чином створюються звіти в БД?
14. Яким чином створюються модулі в БД?
15. Для чого створюють запити?
16. Які режими використовуються для побудови таблиць?
17. Які режими роботи використовуються для роботи із формою?
18. Які режими роботи використовуються для роботи із звітом?
19. Які режими роботи використовуються для роботи із запитом?
20. Які режими роботи використовуються для роботи із макросами?
21. Які режими роботи використовуються для роботи із модулями?
22. Що таке зв'язки між таблицями?
23. Перерахуйте елементи вікна БД?
24. Яким чином можна додавати дані в БД?
25. Яким чином можна створювати об'єкти в БД?
26. Які можна використовувати засоби аналізу в СУБД?
27. Назвіть етапи розробки проекту БД?
28. Назвіть режими створення об'єктів БД?
29. Що таке Поля?

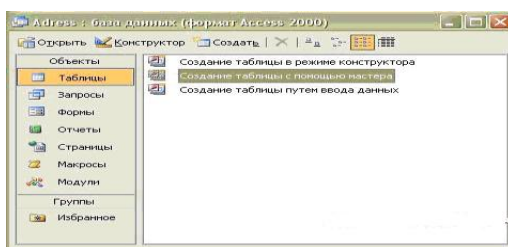
30. Які є типи полів?
31. Назвіть поширені типи полів?
32. Назвіть властивості полів?
33. Назвіть формати полів?
34. Яким чином створюються маски вводу?
35. Призначення маски вводу?
36. Що таке підпис полів?
37. Яке поле називається обов'язковим?
38. Яке поле називається індексованим?
39. Яким чином встановити індекс полів?
40. Яке розширення має БД?

Завдання для самозасвоєння

Завдання 1: Створення бази даних. Знайомство з основними об'єктами бази даних. Створення і заповнення таблиці. Режими подання таблиці. Типи даних. Маска вводу. Зміна структури таблиці в режимі конструктора.

Створення бази даних

1. Заздалегідь створити папку з назвою BAZA.



2. Запустіть MS Access 2003

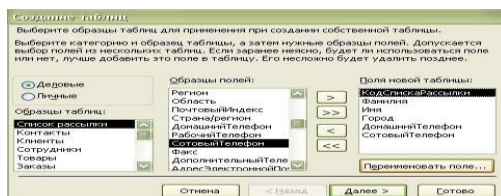
3. Натисніть кнопку **Создать базу даних** на панелі інструментів або в меню

Файл - Створити. Буде відкрито вікно діалогу **Створення бази даних**, до якого слід ввести ім'я бази, наприклад **Address** і вибрати каталог **BAZA**. У результаті ви побачите вікно бази даних.

Створення таблиці за допомогою Майстра

1. У вікні База даних натисніть кнопку **Таблицы** а потім кнопку **Создание таблицы с помощью мастера**

У вікні **Создание таблицы – Мастер таблиц** виберіть зразок **Список рассылки** (самий перший у списку. Далі потрібно визначити перелік полів, які ви хочете включити в таблицю. Для цього по черзі виділяйте мишею поля у списку **Образцов полів** та натисніть кнопку  у щоб зразок поля потрапив до списку полів створюваної таблиці. Виберіть наступні поля: **КодСпискаРассылки**, **Фамилия**, **Имя**, **Адрес**, **Домашний телефон**, **Сотовый телефон**.



2. Перейменуємо назву деяких полів, вибраних із шаблону. Для цього встановлює курсор в вікне **Поля новой таблицы** на рядку **КодСпискаРассылки** і клацніть кнопку **Переименовать поле** і натисніть кнопку **Далее**.

*До тих пір, поки кнопка **Далее** активна, користуйтеся нею, кнопку **Готово** нажимайте тільки в останньому випадку!!!*

3. В наступному вікне діалога введіть ім'я таблиці **Адреса** і дозвольте **MS Access 2003** самостійно встановити ключ і нажміть кнопку **Далее**.

4. В останньому вікні діалога погодьтеся на **Непосредственный ввод данных в таблицу**. І нажміть кнопку **Готово**.

Таблиця створена. В рядку заголовка таблиці можна знайти її назву. Всі імена полів представлені в виді заголовків таблиць.

Змініть заголовок першого поля **КодСпискаРассылки** на **Адресат**.

Заповнення таблиць

В процесі створення таблиці, ви зустрілися з поняттям ключ.

Первинний ключ (в подальшому будемо називати просто ключем) - одне або кілька полів, сукупність яких однозначно визначає будь-яку запис таблиці.

У нашому варіанті ключовим є поле **Адресат**. У таблиці видно, що це поле - **счетчик**, тобто нумерує записи в порядку введення. Заповнювати поле **Адресат** не потрібно, лічильник спрацьовує автоматично, як тільки заповнена хоч одна комірка рядки таблиці.

Заповніть тільки поля **Прізвище та Ім'я**, інші поля залиште порожніми.

Спробуйте заповнити поле **Телефон**.



	Адресат	Фамилия	Имя	Адрес	Домашний телефон	Сотовый телефон
▶	1	Премудрая	Василиса			
	2	Никитич	Добрыня			
	3	Попович	Алеша			
	4	Бессмертный	Кощей			
*	(Счетчик)					

Запись: 1 из 4

Будь-яка таблиця Microsoft Access 2003 може бути представлена в двох режимах:

1. **режиме таблиць**, призначена для вводу даних, їх перегляду і редагування;
2. **режиме конструктора**, призначена для створення і зміни таблиць.

Якщо уважно розглянути вміст осередку, то можна побачити деяку розмітку - маску введення.

Маска вводу - це шаблон, що дозволяє вводити в поле значення, що мають однаковий формат. Маска вводу автоматично зображує в полі постійні символи. При введенні даних у полі немає необхідності набирати ці постійні символи, навіть якщо вони повинні бути включені в значення поля; достатньо заповнити порожні позиції в масці вводу. Крім того, Microsoft Access 2003 не дозволить ввести в поле значення, які не вписуються у визначену для цього поля маску вводу.

Для того, щоб змінити маску введення тексту в телефоні, варто перейти в режим конструктора таблиці. Це можна зробити, вибравши команду

Конструктор меню **Вид** або за допомогою кнопки панелі інструментів .

Вікно конструктора містить таблицю, в першому стовпці якій перераховані поля, а в другому типи даних (формати).

1. Задамо маску вводу для полів **ДомашнійТелефон** і **СотовийТелефон**. Встановіть текстовий курсор в ім'я поля **Домашній телефон** у нижньому полі Загальні навпроти **Маски введення** і створіть шаблон у вікні **Создание масок** вводу клацніть **Список** введіть у полі **Опис** - Номер телефону, **Маска вводу** - 0 00 00, **Зразки даних** - 2-66-34 та клацніть по кнопці **Вперед** на 1 запис. Створіть маску вводу для номера стільникового телефону: **Опис** - Номер телефону, **Маска вводу** - 000-000-00-00, **Зразок даних** - 902-275-78-33 клацання по ► і **Закрити**.

2. У вікні **Создание масок** вводу клацніть по полю **Номер телефону 2-66-34** - Далі - Готово.

3. Натиснувши на рядку **Конструктора Мобільний телефон**, змініть для нього маску введення на десятизначний номер зі створеного вами шаблону. Перейдіть з режиму **Конструктора** в режим **Таблиці**, для цього виконайте

команду **Вид - Таблица** або натисніть кнопку на панелі інструментів  . На питання про збереження таблиці відповідайте **Да**. Далі заповнення номерів телефонів полягає у введенні в кожну клітинку цифр, переміщує курсор з комірки в комірку.

4. Поверніться до таблиці і заповніть поле **Адреса**. Введіть дані в полі **Адреса**. Змініть ширину стовпців таблиці за аналогією з електронними таблицями. Приблизний вигляд таблиця показаний на малюнку нижче.



	Адресат	Фамилия	Имя	Адрес	Домашний телефон	Сотовый телефон
1	Премудрая	Василиса		Тридешатое шоссе, 24	2-67-89	902-567-37-58
2	Никитич	Добрыня		Рязанская ул., 333	6-89-56	912-564-78-00
3	Попович	Алеша		Муромский пер., 111	2-34-56	908-477-71-10
4	Бессмертный	Кощей		Тридешатый проспект, 666	3-45-56	905-788-99-06

Запись: 14 из 5

Завдання 2: Створення форми. Введення записів у форму. Режими подання форми. Швидке сортування даних.

Створення форми

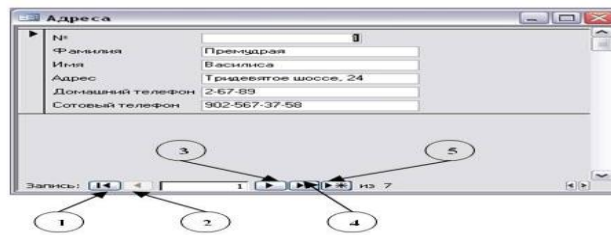
1. Запустіть MS Access 2003 і відкрийте створену на минулому занятті базу даних.

2. Для створення форми клацніть на ярличку **Форми** на панелі Об'єкти та клацніть на кнопці **Створення форми** за допомогою **Майстра** в лівому вікні.

3. У вікні Створення форм, опція **Таблиці** та запити доступна Таблица: **Адреси**. У вікні Доступні поля зі списку всі доступні поля перемістити в праве вікно Вибрані поля і натиснути кнопку **Далі**. У наступному вікні виберіть пункт в один стовпець - **Далі**. Виберіть стиль: **Стандартний - Далі**.

4. Назва форми - **Адреси**. Подальші дії: **відкрити форму для перегляду та введення даних - Готово**.

5. На екрані ви побачите приблизно наступне:

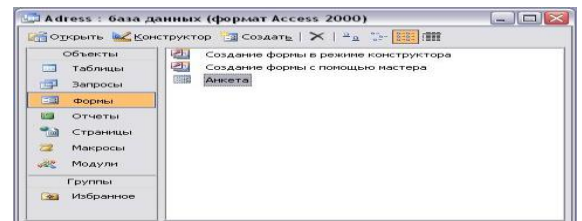


Представлення записів у формі

Кожен запис (те, що було окремим рядком таблиці) тепер представляє як би окрему картку. Вводити дані зручно саме в такому режимі.

У рядку стану (в нижній частині вікна форми) відображено загальна кількість записів і номер запису, який ви зараз бачите.

Переміщатися між записами можна за допомогою кнопок, значення яких показано на малюнку зліва. 1 - до останнього запису, 2 - на один запис назад, 3 - на один запис вперед, 4 - до останнього запису, 5 - новий запис.



1. Перегорніть всі наявні записи (перегляньте картки), використовуючи кнопки для переміщення по формах.

2. Внесіть нового адресата (запис 5): **Прізвище - Горинич, Ім'я - Змій, Адреса - Чорна гора, 13, Домашній телефон - 2-34-55, Мобільний телефон - 905-678-00-11**

3. Перейменуйте поле **Адресат** в полі на **№**. Перейдіть з Форм в **Таблицю**, для цього закрийте форму **Адреси** та відкрийте **Таблицю Адреси**. Клацніть на заголовку поля **Адресат** і перейменуйте його. Змініть ширину стовпця **№** відповідно до розміру заголовка.

4. Змініть ім'я форми **Адреси** на **Анкета** по завершенню операції ви побачите наступне

Форму можна представляти в режимі **таблиці**. Для цього запустіть форму **Анкета** і виконайте команду **Вид - Таблица**.

У режимі **таблиця** можна вносити та редагувати дані. Можна видаляти записи. У таблиці існує можливість переставляти стовпці. Зробити це можна, схопивши стовпець за заголовок мишею, перенести його на нове місце.

5. Додайте нові записи в таблицю, змініть запис під номером 3, так само змініть порядок стовпців, як показано на малюнку.

Адреса : таблица						
	№	Фамилия	Имя	Домашний телефон	Сотовый телефон	Адрес
	1	Премудрая	Василиса	2-67-89	902-567-37-58	Тридевятиое шоссе, 24
	2	Никитич	Добрыня	6-89-56	912-564-78-00	Рязанский ул. , 333
	3	Муромец	Илья	2-34-56	908-477-71-10	Муромский пер. , 111
	4	Бессмертный	Кощей	3-45-56	905-788-99-06	Тридесатый проспект, 666
	5	Горыныч	Змей	2-34-55	905-678-00-11	Черная гора, 13
	6	Крошечка	Хаврошечка	6-32-11	955-087-65-43	Яблоневый проезд, 98
	7	Прекрасная	Василиса	6-65-43	909-654-32-13	Тридевятиое шоссе, 45
►	чик)					

1. Записи в таблиці можна відсортувати. Для цього:

- виділити стовпчик, по якому буде проходити сортування (в режимі форми виділеного поля);
- виберіть в меню Записи команду **Быстрая сортировка** і **По возрастанию** або **По убыванию** в підменю. Можна натиснути одну із кнопок панелі інструментів



По возрастанию



По убыванию.

2. Для того щоб повернутися в режим форми потрібно виконати команду **Вид – Режим Формы**

3. Відсортируйте записи таким чином, щоб імена розміщувались по алфавіту. Переключитесь в режим **Формы** і перевірте, зберігається чи режим записів.

Завдання № 3. Створення звітів

1. В вікні **База данных** активізуйте вкладку **Отчеты** і клацніть кнопку **Создать**.

2. З допомогою **Мастера отчетов** створіть звіт для виводу відомостей про адресатів, вибрати для звіту наступні поля: *Адресат, Фамилия, Имя, Мобільний телефон*.

В якості джерела даних використовуйте таблицю *Адреса*. При створення звіту використовувати сортування по полю *Фамилия*, вид звіту *Табличный*, стиль *Строгий*.

Ввести ім'я звіту (по замовчуванню Access вводить ім'я таблиці-джерела).

3. Закрити звіт і вийти в вікна **База данных**.

Завдання 4: Створення Запитів

Створити запит-вибірку з таблиці Адреси, що містить відомості про адресатів:

1. У вікні База_даних активізувати вкладку Запрос і натиснути кнопку Створити.
 2. Обрати режим Конструктор і підтвердити вибір.
 3. У вікні Додавання таблиці виділити ім'я таблиці, з якої проводитиметься запит (Адреси) і виконати команду Додати. Список полів цієї таблиці повинен з'явитися у вікні Запрос_на_виборку.
 4. Закрити вікно **Добавление_таблицы**.
 5. У вікні, що залишився **Запрос_на_виборку** клацнути верхню ліву комірку, що відноситься до заголовка Поле.
 6. У даній комірці повинен з'явитися значок спливаючого меню.
Використовуючи його, ввести в клітинку ім'я першого поля створюваного запиту (наприклад, Прізвище).
 7. Аналогічним чином заповнити інші комірки першого рядка (Прізвище, Ім'я, Адреса, Мобільний телефон).
 8. Встановити сортування по полю Прізвище.
 9. У рядку **Умови відбору** внести в потрібні поля критерії відбору: у полі **Пол** встановити букву **ж**, (дані для умови відбору вносити до такого ж вигляді, як вони внесені в таблицю).
 10. У рядку Вивод__на_екран значок «галочка» означає, що в результаті виконання запиту на цьому полі буде виводитися на екран.
 11. Виконати команду Запит \ Запуск або використовувати відповідну піктограму.
- Багато табличних БД, встановлення зв'язків між таблицями.

12. Переглядаємо кожну із створених об'єктів, вносимо, якщо необхідно певні зміни.
13. Виконану роботу показати викладачу.
14. Закрити всі вікна та виключити ПК.

Список використаної літератури

1. Інформатика. Комп'ютерна техніка. Комп'ютерні технології: Підручник. 3-є вид./ За ред. Г.А. Шинкаренка, О.В. Шишова. – К.: Каравела, 2008. – 640с.
2. Злобін Г.Г. Основи інформатики, комп'ютерної техніки і комп'ютерних технологій (для студентів економічних спеціальностей) /Г.Г. Злобін.- К.: Каравела, 2008.– 240с.
3. Злобін Г.Г. Персональний комп'ютер: Навч. Курс. 2-е вид /Г.Г. Злобін.- Львів.: Простір М, 2007.- 294с.
4. Баузр Ф.Л. Информатика. Вводный курс /Ф.Л. Бауз.- М.: Мир, 2004. -336с.
5. Галузинський Г.П. Сучасні технологічні засоби обробки інформації: Навчальний посібник /Г.П. Галузинський, І.В. Гордієнко.- К., 2004. - 224 с.
6. Денисов А.Н. Интернет. Самоучитель /А.Н. Денисов, И.Н. Вихарев, А.В. Белов.- СПб.: Питер, 2001. - 464 с.
7. Дибкова Л.М. Інформатика та комп'ютерна техніка. Посібник для студентів вищих навчальних закладів /Л.М. Дибкова.- К.: Видавничий центр “Академія”, 2002.-320с.
8. Ефимова О. В. Практикум по компьютерной технологии /О.В. Ефимова.- М.: АБО, 2003. - 432 с.
9. Інформатика. Комп'ютерна техніка. Комп'ютерні технології: Підручник.- К.: Каравела, 2004.- 464 с.
10. Інформатика і комп'ютерна техніка: Навч. посіб. для дист. навчання. / Уклад.: Наумова Н.М. та ін. – К.: Університет «Україна», 2004.-404 с.
11. Інформатика: Комп'ютерна техніка. Комп'ютерні технології: Підручник для студентів вищих навчальних закладів/За ред. О.І.Пушкаря.-К.: Видавничий центр “Академія”, 2003.-704с
12. Руденко В.Д. Практичний курс інформатики /В.Д. Руденко, О.М. Макаручук, М.О. Патланжоглу.-К.: Фенікс, 2001. - 370с.
13. Информатика: Базовый курс/С.В.Симонович и др. -СПб.: Питер, 2003. - 640 с.

14. Наумова Н.М. Інформатика і комп'ютерна техніка: Навч. посіб. для дист. навчання. / Наумова Н.М.- К.: Університет «Україна», 2004.-404 с.
15. Шинкаренко Г.А. Інформатика. Комп'ютерна техніка. Комп'ютерні технології / Г.А. Шинкаренко, О.В. Шишов. – К.: Каравела, 2007. – 640с.

ЗМІСТ

Вступ.....	3
1. Основні концепції бази даних.....	4
2. Проектування бази даних.....	6
3. Microsoft Access як реляційна СУБД.....	13
3.1. Визначення даних і їх збереження.....	13
3.2. Обробка даних.....	15
4. Архітектура Microsoft Access.....	18
5. Таблиці і поля.....	19
5.1. Стовпчики і рядки, поля та записи.....	20
5.2. Робота з вікном бази даних.....	21
5.3. Створення нової таблиці.....	22
5.4. Типи полів.....	24
5.5. Використання засобу «Описание».....	25
5.6. Властивості поля.....	25
5.7. Формати полів (Format).....	26
5.8. Маска введення (Input Mask).....	27
5.9. Підпис поля (Caption).....	27
5.10. Значення за замовчування (Default Value).....	27
5.11. Умова на значення (Validation Rule) і повідомлення про помилку (Validation Text).....	28
5.12. Обов'язкове поле (Required).....	29
5.13. Порожні рядки (Allow Zero Length).....	29
5.14. Індексване поле.....	30
5.15. Установка індексів — первинних ключів.....	31
Список використаної літератури.....	42

НАВЧАЛЬНО-МЕТОДИЧНЕ ВИДАННЯ

Методичний посібник для самостійного вивчення з курсу " СУБД Microsoft Access " для студентів 1 курсів напрямків «Здоров'я людини», «Правознавство», «Маркетинг», «Туризм»

денної та заочної форм навчання

Комп'ютерний набір та верстка:

Л.А. Сидорчук, І.М. Поліщук

Підп. до друку _____ 2010. Формат 60×84/16. Папір офс.

Гарн. Таймс. Ум друк. арк. 3

Обл.-вид.арк. 2,5. Тираж 10 прим. Зам.3852

Луцький інститут розвитку людини

Університету «Україна»

43023, м. Луцьк, вул. Карбишева, 2