МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЦЕНТРАЛЬНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Механіко-технологічний факультет
Кафедра кібербезпеки та програмного забезпечення

МЕТОДИЧНІ ВКАЗІВКИ

до виконання лабораторних робіт з навчальної дисципліни «Організація баз даних» (2 частина) для студентів денної та заочної форми навчання за спеціальностями 6.050102/123 «Комп'ютерна інженерія», 125 «Кібербезпека»

ЗАТВЕРДЖЕНО
кафедрою кібербезпеки та
програмного забезпечення,
протокол від 5 липня 2017 року № 1

КРОПИВНИЦЬКИЙ
2017

Методичні вказівки до виконання лабораторних робіт з навчальної дисципліни «Організація баз даних» (2 частина) для студентів денної та заочної форми навчання за спеціальностями 6.050102/123 «Комп'ютерна інженерія», 125 «Кібербезпека» / уклад. В.В. Сидоренко, Л.В. Константинова — Кропивницький: ЦНТУ, 2017. — 41с.

Укладач: Сидоренко В. В., Константинова Л.В.

Рецензенти: Смірнов О. А., д-р техн. наук, професор;
Дрєєв О. М., канд. техн. наук.

Вступ

В методичних вказівках представлено теоретичний і практичний матеріал для роботи з базами даних за допомогою СКБД Microsoft Access. Дані методичні вказівки продовжують методичні вказівки до лабораторних робіт з навчальної дисципліни «Організація баз даних» (1 частина) для студентів денної та заочної форми навчання. Приводиться багато завдань і практичних прикладів та ілюстрацій, які допомагають краще засвоїти викладений матеріал з дисципліни «Організація баз даних». Розглядається робота з БД за допомогою запитів, основні прийоми роботи з БД за допомогою мови SQL, робота з макросами, застосування VBA для розробки додатків користувачів, захист бази даних.

Теми лабораторних робіт, що розглядаються та оцінювання знань

Пропонується виконати лабораторні роботи за наступними темами:

Теми лабораторних робіт	Години		
	Д.ф.	З.ф	З.ф.
1. Керування базами даних за допомогою SQL. Вибірка, читання даних, отримання підсумкових значень. Однотабличні запити. Внесення змін до баз даних.	2	1	1
2. Керування базами даних за допомогою SQL. Особливості багатотабличних запитів. Застосування вкладених запитів. Зміна структури БД за допомогою операторів DDL. Створення та вилучення таблиць. Створення та видалення унікальних індексів.	2	1	1
3. Проектування і використання баз даних. Автоматизація роботи з базою даних за допомогою макросів.	2	1	1
4. Проектування і використання баз даних. Розробка додатків користувачів за допомогою VBA.	4	1	1
5. Зберігання інформації у базах даних. Захист бази даних, OLE-об'єкти у БД.	2		
6. Інформаційні системи в мережах. Зв'язок БД та електронних таблиць. Встановлення зв'язків та публікація даних у Web.	2		
Всього:	14	4	4

Форма підсумкового контролю іспит.

Максимальну кількість балів студент може одержати у випадку відвідування всіх лекцій, лабораторних занять, виконання і захисту завдань з самостійної роботи у встановлений термін проходження контролю.

При виконанні і при захисті лабораторних робіт після встановленого терміну, одержані бали перераховуються з коефіцієнтом: для самостійної роботи студента -0,3; лабораторної роботи -0,7.

Шкала оцінювання: національна та ECTS

Сума балів за всі види навчальної діяльності	Оцінка ECTS	Оцінка за національною шкалою
		для екзамену, курсової роботи
90 – 100	**A**	відмінно
82-89	**B**	добре
74-81	**C**	
64-73	**D**	задовільно
60-63	**E**	
35-59	**FX**	незадовільно з можливістю повторного складання
0-34	**F**	незадовільно з обов'язковим повторним вивченням дисципліни

Вибравши предметну область, над якою будете працювати, ви повинні виконати завдання до лабораторних робіт, а також відповісти на питання в кінці кожної лабораторної роботи. Звіт повинен містити хід виконання завдань а також графічні матеріали, що підтверджують виконання цих завдань.

Лабораторна робота №1

Тема: Керування базами даних за допомогою SQL. Вибірка, читання даних, отримання підсумкових значень. Однотабличні запити. Внесення змін до баз даних.

Мета: Застосовуючи SQL-оператори, та спеціальні засоби навчитися виконувати складну обробку даних за допомогою запитів та вдосконалювати їх виведення. Навчитися виконувати узагальнену групову обробку значень полів за допомогою агрегатних функцій в SQL запитах. Навчитися застосовувати засоби, які керують значеннями в таблицях. Засвоїти команди DML. Навчитись за допомогою SQL – запитів модифікувати, видаляти, вносити інформацію в БД.

Зміст роботи за варіантом індивідуального завдання:

1. За допомогою SQL- запиту вивести всі дані з певної таблиці. В інструкції застосувати необов'язкове скорочення у вигляді символа «зірочка» (*).
2. Вивести за допомогою SQL- запиту інформацію з таблиці тільки за двома полями.
3. Вивести дані поля вашої таблиці таким чином, щоб результат не мав дублікатів (створити SQL- запит).
4. За таблицею з даними отримайте інформацію про всі об'єкти (постачальники, клієнти, хворі, студенти і т.д.), яким в полі, наприклад, код або номер відповідає певне значення (наприклад 2003), а в іншому полі- значення більше або дорівнює певному значенню (наприклад>=3).
5. За допомогою агрегатної функції у SQL- запиті підрахуйте кількість записів у таблиці вашої БД, не рахувати пусті значення, але враховувати дублікати.
6. Необхідно знайти максимальне значення збільшеного вдвічі значення поля, наприклад стипендії або заробітної плати, або ціни товару, або іншого поля зі своєї БД за допомогою SQL- запиту.
7. Отримайте інформацію з таблиці з даними про студентів (чи інші об'єкти), впорядковуючи їх за розміром стипендії (або інше числове поле) у порядку спадання, а для студентів, що мають однаковий розмір стипендії в алфавітному порядку їх прізвищ.
8. За допомогою DML у вікні SQL побудувати запит, завдяки якому можна було б вносити нові записи до вашої таблиці. Два поля таблиці повинні заповнюватись певною інформацією, третє поле повинно мати значення NULL, а останні -за замовчанням.
9. Вилучіть з спеціально для цього створеної таблиці всі записи за допомогою SQL.
10. Вилучіть з таблиці ті записи, які у конкретному полі мають певне значення за допомогою DML.
11. За допомогою DML змініть значення де-якого поля на «пусто», якщо поле дата має значення 01.01.2010.
12. Зменшити за допомогою DML значення числового поля на 25% в таблиці, якщо, його значення, наприклад, дорівнює 32500.

13. В режимі **SQL Вид** створіть інструкцію, за допомогою якої створюється копія деякої таблиці.

Теоретичні відомості

Запит SQL— це запит, який створюється за допомогою інструкцій SQL. Інструкція SQL це вираз, який визначає команду SQL (SELECT, UPDATE або DELETE), та включає рядки (наприклад WHERE або ORDER BY). Інструкції SQL зазвичай застосовуються в запитах та в статистичних функціях. Мова SQL (Structured Query Language) застосовується при створенні запитів, а також для оновлення та керування реляційними базами даних, такими як бази даних Microsoft Access.

SQL - запит складається з послідовності SQL - інструкцій (SQL-statement), які вказують, що потрібно зробити з вхідним набором даних (таблицею або запитом) для генерації вихідного набору. Шляхом аргументів (параметрів) цих інструкцій (clause) конкретизуються дії, що виконуються, тобто задають імена полів, імена таблиць, умови, відношення і т.п. Витягнута з бази інформація (вихідний набір) обробляється за допомогою спеціальних статистичних функцій (aggregate functions). При такій обробці можна визначити, наприклад, мінімальне і максимальне значення, суму і середнє значення.

Інструкції SQL можна застосовувати в Microsoft Access в тих місцях, де можливо ввести ім’я таблиці, запиту або поля.

В деяких ситуаціях інструкція SQL створюється автоматично. Наприклад, застосовуючи майстер для створення форми чи звіту, які отримують дані з декількох таблиць, автоматично створюється інструкція SQL, яка буде значенням властивості **Источник строк** (RowSource) форми чи звіту. Створюючи список чи поля зі списком за допомогою майстра також створюється інструкція SQL, яка стає значенням властивості **Источник строк** (RowSource) списку чи поля із списком.

Якщо майстер не застосовувався, можливо створити інструкцію SQL у комірках властивостей **Источник записей** (RecordSource) або **Источник строк** (RowSource), застосовуючи побудовувач поряд з відповідною коміркою та створивши запит в режимі конструктора запиту.

За допомогою мови структурованих запитів SQL, що реалізована в Access, можна скласти будь-яку кількість складних запитів. Ця мова дозволяє також управляти обробкою запитів, SQL-запит являє собою послідовність інструкцій, в яку можуть входити вирази і статистичні функції SQL.

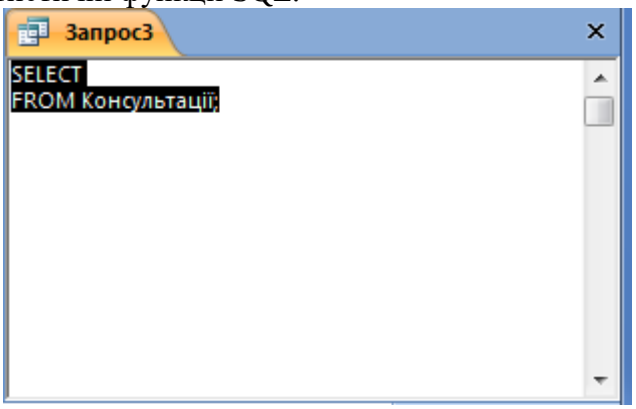

Рисунок 1

Коли в режимі конструктора користувач створює специфікацію запиту, Access водночас будує відповідний SQL-запит. Зміни в SQL-запиті автоматично

відображаються і в специфікації QBE - запиту. Щоб відобразити на екрані або виправити SQL-запит, потрібно викликати команду **SQL Вид** з меню **Вид** (в режимі конструктора запиту) рисунок 1.

В вікні **SQL** вводяться інструкції, що складають SQL-запит. При введенні тексту в цьому вікні занадто довгі рядки розриваються. Для підвищення наочності інструкції кожний командний рядок SQL можна починати з нового рядка, використовуючи комбінацію клавіш (Ctrl+Enter). Наприклад:

```
SELECT SNUM, SFAM, SIMA, SOTCH, STIP
FROM STUDENTS;
```

SELECT- ключове слово, яке повідомляє БД, що ця команда є запитом.

SNUM, SFAM, SIMA, SOTCH, STIP-список полів з таблиці, які вибираються запитом. Такий запит не впливає на інформацію в таблицях, він тільки показує дані.

FROM STUDENTS – ключове слово, яке супроводжується ім'ям таблиці, яку використовуємо у якості джерела інформації.

Крапка з комою «;»-застосовується у всіх інтерактивних командах SQL для повідомлення БД що команда готова для виконання.

«*» -якщо необхідно отримати кожне поле таблиці. Наприклад:

```
SELECT * FROM STUDENTS;
```

що призведе до того ж результату, що і попередня команда. Якщо необхідно вивести тільки деякі поля з таблиці, просто необхідно виключити з списку непотрібні поля.

Якщо є необхідність уникнути дублювання застосовують DISTINCT-аргумент. Наприклад:

```
SELECT DISTINCT SNUM FROM USP;
```

Цю інструкцію застосовують для отримання списку результатів без дублікатів.

Якщо застосувати ALL, то це буде мати протилежний ефект.

WHERE – інструкція команди SELECT, яка дозволяє встановлювати предікати, умова яких можливо буде виконуватись або не виконуватись для будь-якого запису таблиці. Команда отримує тільки ті записи з таблиці, для яких таке твердження буде вірним. Наприклад:

```
SELECT SFAM, STIP
FROM STUDENTS
WHERE STIP=25.50;
```

виведуться тільки ті прізвища та розмір стипендії студентів, для яких у полі STIP буде значення 25.50.

Пов'язуючи предікати з булєвськими операторами (AND, OR, NOT) та реляційними операторами (=, >, <, >=, <=, <>), набагато збільшується можливість вибірки потрібних даних. Також можливо застосовувати спеціальні оператори IN, BETWEEN, LIKE, IS NULL.

Запити здатні виконувати узагальнену групову обробку значень полів, що реалізовується за допомогою агрегатних функцій. Агрегатні функції отримують одиночне значення для всієї групи таблиці. В SQL допускають наступні функції:

COUNT, SUM, AVG, MAX, MIN. Щоб знайти суму всієї отриманої стипендії в таблиці про студентів маємо:

```
SELECT SUM (STIP)
FROM STUDENTS;
```

Функція COUNT застосовується для підрахунку кількості значень у стовпці. Наприклад:

```
SELECT COUNT (SNUM)
FROM USP;
```

Якщо з COUNT застосовувати ALL (використовується за замовчанням) – не можна підрахувати значення NULL, але враховуються дублікати, а COUNT з «*» включає записи з NULL та дублікати.

Для впорядкування виведення полів таблиць SQL має команду ORDER BY, що дозволяє сортувати виведення запиту згідно зі значеннями серед кількості полів. Якщо вказати декілька полів, то стовпці виведення впорядковуються один в середині іншого, при цьому треба визначити зростання (ASC) чи спадання (DESC) для кожного стовпця. Наприклад:

```
SELECT *
FROM USP
ORDER BY SFAM ASC;
```

виведе таблицю з інформацією про студентів в алфавітному порядку прізвищ.

Однотабличні запити – це запити, які в якості джерела даних використовують тільки одну таблицю.

Якщо в SQL-запиті немає помилок, Access сформує еквівалентний QBE-запит і заповне поля QBE-області. Тепер запит можна виконати, викликавши команду **Выполнить** або активізувавши кнопку із знаком оклику на панелі інструментів.

Прикладами SQL-запитів є запити на об'єднання, запити до сервера, керуючі і підлеглі запити і т.д.. Деякі SQL-запити неможливо створити у бланку запиту, для визначення запиту у режимі конструктора.

Запит до сервера-запит SQL, який застосовують для передачі команд безпосередньо на сервер бази даних ODBC. Запрос до сервера дозволяє безпосередньо працювати з таблицями на сервері замість обробки їх даних за допомогою ядра бази даних Microsoft Access.

Керуючий запит- запит SQL, який вміщує інструкції мови опису даних DDL. Такі інструкції дозволяють створювати або змінювати об’єкти у БД.

Запит на об’єднання-запит, в якому за допомогою оператора UNION об’єднуються результати двох або декількох запитів на вибірку.

Підлеглі запити-запити, в яких інструкция SQL SELECT, вміщується всередині іншого запиту на вибірку чи на зміну.

Розробка запитів на об'єднання, запитів до сервера і керуючих запитів здійснюється тільки в вікні запиту в режимі SQL, а для підлеглих запитів користувач може ввести інструкцію SQL у рядок **Поле** або **Условие отбора** в бланку запиту.

Запити на зміну за допомогою SQL

Під час роботи з SQL важливо вміти користуватися засобами керування інформацією у таблицях. Як ви вже знаєте, запити на зміну використовуються для додавання, вилучення і поновлення записів, а також для збереження результуючого набору записів запиту у вигляді таблиці. Значення можуть вставлятись, модифікуватись та видалятись з полів за допомогою трьох команд **DML (Мова маніпулювання даними) - INSERT, UPDATE, DELETE.**

Існує чотири підвиди запитів на зміну, кожному з яких відповідає певна SQL інструкція.

Таблиця – Види запитів на зміну та їх інструкції

Запрос дії	**Опис**	**SQL –інструкція**
Append query	Запит на додавання записів	INSERT INTO
Delete query	Запит на вилучення записів	DELETE
Make-table query	Запит на створення таблиці	SELECT ...INTO
Update query	Запит на поновлення записів	UPDATE

Додавання інформації в БД

В SQL всі записи в таблицю вводяться за допомогою команди модифікації INSERT. Але слід пам'ятати, що ім'я таблиці, в яку відбувається вставка, повинно бути попередньо визначене, а кожне значення, що вставляється повинне співпадати з типом даних стовпця, в який воно вставляється. Наприклад, для додавання запису в таблицю викладачів TEACHERS, можна скористатись наступним виразом:

```
INSERT INTO TEACHERS
VALUES (4006, "Федченко", "Світлана","Геннадієвна", #01/09/1999#);
```

Можна вказати стовпці, в які необхідно здійснити вставку значень. Наприклад:

```
INSERT INTO TEACHERS (TDATE, TFAM, TIMA)
VALUES (#01/09/1999#, "Федченко", "Світлана");
```

Для полів, які не вказані в запиті автоматично встановлюються значення за замовчанням.

Є можливість за допомогою команди INSERT отримувати чи вибирати значення з однієї таблиці та поміщати їх в іншу разом з запитом.

Видалення даних

Видалення рядків з таблиці можна здійснити командою модифікації DELETE. Треба враховувати, те, що команда може видаляти тільки цілі записи таблиці, а не індивідуальні значення де-яких полів. Для видалення всього вмісту таблиці STUDENTS можливо виконати наступне:

```
DELETE FROM STUDENTS;
```

Для визначення рядків, які треба видалити за умови, застосовують предікат. Наприклад, для видалення інформації, що стосується студента Нагорного можна використати наступну команду:

```
DELETE FROM STUDENTS
WHERE SNUM=3416;
```

В якості предікату використовують номер студентського квитка.

Зміна існуючих даних

Можливість зміни всіх або деяких значень в таблиці реалізується за допомогою команди UPDATE. В ній вказується ім'я таблиці, яка використовується та слово SET, яке визначає зміну яка відбудеться для потрібного поля таблиці. Наприклад:

```
UPDATE USP
SET OCENKA=5;
```

Приведе до зміни в таблиці USP всіх оцінок на «5».

Для зміни єдиного значення можна застосовувати предікати. Наприклад:

```
UPDATE USP
SET OCENKA=5;
WHERE PNUM=2003;
```

За допомогою UPDATE можна модифікувати дані з декількох полів- SET взмозі визначити будь-яку кількість полів, відокремлених комами.

Модифікувати зразу кілька таблиць однією командою UPDATE не дозволено, тому не можна застосовувати назву таблиці через крапку в іменах полів

У рядку SET команди UPDATE можна застосовувати вирази, розташовуючи їх в списку для того поля, яке необхідно змінити. Наприклад:

```
  UPDATE STUDENTS
  SET STIP=STIP*2;
```

Команда збільшує значення стипендії в 2 рази.

До недоліків команди UPDATE можна віднести неможливість посилатися на таблицю, яка задіяна в будь-якому підзапиті з команди модифікації. Наприклад, неможливо одною командою виконати таку дію, як модифікація оцінок для студентів, у яких оцінки нижче середньої.Для цього необхідно виконати один запит:

```
SELECT AVG (OCENKA)
FROM USP;
```

а потім результат цього запиту застосувати для модифікації:

```
UPDATE USP
SET OCENKA= OCENKA-1
WHERE OCENKA <4.2;
```

Запит на створення таблиці

За допомогою SQL-інструкції можна створювати таблиці на основі вже існуючих таблиць. Наприклад**:**

```
SELECT * INTO VIPUSK
FROM STUDENT;
```

За допомогою цих інструкцій буде створена таблиця VIPUSK, яка містить всі дані з таблиці STUDENT.

Перетворення QBE - запиту в SQL-запит

Запити, створені в вікні конструктора запиту, Access автоматично перетворює в SQL запити.

SQL-запит можна відредагувати в вікні **SQL.** Після закриття вікна зміни автоматично відображаються і в QBE-запиті.

Контрольні питання до лабораторної роботи 1:

1. **Які запити називають SQL-запитами?**
2. **Яке головне призначення SQL-запитів?**
3. **За допомогою якої інструкції в запитах здійснюється вибірка даних?**
4. **За допомогою якого символу в інструкції SQL-запитів можна вивести всі дані з таблиці?**
5. **Який засіб запобігає дублюванню інформації в SQL-запитах?**
6. **В яких ситуаціях інструкція SQL створюється автоматично в Microsoft Access?**
7. **За допомогою якого засобу у SQL-запитах в результаті зберігається дублювання рядків виведення?**
8. **Що означає інструкція SELECT...FROM...?**
9. **Для чого застосовується команда ORDER BY?**
10. **Яка інструкція команди SELECT, дозволяє встановлювати предікати?**
11. **Яка функція застосовується для підрахунку кількості значень у стовпці таблиці в SQL-запиті?**
12. **Які особливості мови SQL?**
13. **Чи можливо перетворювати SQL запити в QBE і навпаки?**
14. **Які запити на зміну ви знаєте?**
15. **За допомогою якої інструкціі можна створити запит на додавання записів у таблицю?**
16. **Що означає команда UPDATE у запитах?**
17. **Яку інструкцію SQL повинен мати запит на вилучення записів?**
18. **Що означає інструкція SELECT ...INTO в запитах?**

Лабораторна робота №2

Тема: Керування базами даних за допомогою SQL. Особливості багатотабличних запитів. Застосування вкладених запитів. Зміна структури БД за допомогою операторів DDL.Створення та вилучення таблиць.Створення та видалення унікальних індексів.

Мета: Застосовуючи SQL-оператори EXISTS, ANY, ALL, SOME навчитися структурувати запити, створювати більш зрозумілі запити. Навчитися будувати вкладені запити. Засвоїти на практиці операції об'єднання. Застосовуючи DDL оператори CREATE, DROP, ALTER навчитися створювати та визначати, видаляти та змінювати об'єкти БД.

Зміст роботи за варіантом індивідуального завдання:

1. За допомогою SQL- запиту вивести дані з двох таблиць, наприклад, про викладачів і студентів, чиї фамілії знаходяться між буквами 'К' і 'С', крім того зазначити, хто студент, а хто викладач.
2. Вивести за допомогою SQL-запиту інформацію з таблиць, наприклад, викладачів і предметів (які пов'язані між собою), щоб кожному викладачу відповідав предмет, який він викладав.
3. Необхідно отримати інформацію, наприклад, про студента Петрова-номер у списку та бал успішності. Номер знаходиться у таблиці про студентів, бал- у таблиці успішності. Застосуйте вкладений запит.
4. Вивести з таблиці успішності за допомогою SQL- запиту номера студентських білетів та код предмету, тільки якщо в ній присутні відмінні оцінки.
5. За допомогою DDL-операторів створіть таблицю з визначеним ім'ям та декількома полями різних типів та розмірів.
6. За допомогою DDL-оператора змініть структуру існуючої таблиці. Додайте нове поле типу INTEGER, збільшіть розмір існуючого поля, видаліть непотрібне поле.
7. Видаліть таблицю із своєї БД за допомогою DDL-операторів.
8. Створіть індекс у таблиці БД.
9. Створіть унікальний індекс для таблиці своєї БД.
10. Видаліть створений раніше індекс.
11. Створіть таблицю так, щоб для двох полів не можна було б встановити невизначене значення, та встановіть первинний ключ за допомогою необхідного обмеження.

Теоретичні відомості

Часто виникає необхідність у виборі інформації з декількох таблиць. Що вирішується за допомогою багатотабличних запитів. Одним із варіантів такого виводу є об'єднання результатів декількох запитів, які виконуються незалежно один від одного.

Для розташування декількох запитів разом та об'єднання їх виведення застосовують UNION, що поєднує виведення двох чи більше SQL запитів в єдиний

набір рядків та стовпців. Наприклад, щоб отримати інформацію про учасників конференції ВНЗ застосовується запит:

```
SELECT SFAM, SIMA, SOTCH
FROM STUDENTS
WHERE NOT(ISNULL(KONF))
UNION
SELECT SFAM, SIMA, SOTCH
FROM TEACHERS
WHERE NOT(ISNULL(KONF));
```

Зверніть увагу, відсутність «;» після першої інструкції означає, що далі буде ще один або декілька запитів. Якщо необхідно з'єднати два або більше стовпців враховується, що вони повинні бути сумісними для об'єднання. Для кожного з запитів необхідне включення однакової кількості стовпців, у тому ж порядку, та при цьому повинна бути сумісність типів. Не можна застосовувати агрегатні функції в інструкції SELECT запиту на об'єднання. UNION автоматично виключає дублікати рядків з виведення. Можливо застосовувати константи та вирази у інструкції SELECT з UNION. Наприклад:

```
SELECT "Студент  ", SFAM
FROM STUDENTS
UNION
SELECT "Викладач  " TFAM
FROM TEACHERS;
```

Результат на рис. 1.

Рисунок 1

Важливою особливістю запитів SQL є їх спроможність визначати зв'язки між таблицями і виводити інформацію з них в термінах цих зв'язків. Ці операції називаються об'єднанням. У багатотабличних запитах, таблиці, які представлені у вигляді списку в інструкції FROM, відокремлюються одна від одної комами. Предикат запиту може посилатись на будь-який стовпець будь-якої таблиці, та може використовуватись для зв'язку між ними. Тепер виникає необхідність вживання імен стовпців та таблиць, оскільки в багатотабличному запиті можливе виникнення неоднозначності. Допускається створювати запити, що об'єднують більше двох таблиць. Наприклад, необхідно вивести список оцінок, які виставив той чи інший викладач:

```
SELECT TEACHERS.TFAM, USP.OCENKA
FROM TEACHERS, PREDMET, USP
WHERE TEACHERS.TNUM=PREDMET.TNUM
AND PREDMET.PNUM=USP.PNUM;
```

Запити здатні керувати іншими запитами, це відбувається, якщо розмістити запит всередину іншого предиката, який використовує виведення внутрішнього запиту для встановлення вірного чи невірного значення предиката. Наприклад, потрібно вивести

всю інформацію про студента з прізвищем Поляков, якщо невідомий його номер. Номер отримується з таблиці з даними про студентів, а потім застосувати результат до таблиці успішності таким запитом:

```
SELECT *
FROM USP
WHERE SNUM=
(SELECT SNAM
FROM STUDENTS
WHERE SFAM=‘Поляков’);
```

Щоб виконати запит SQL спочатку оцінює внутрішній запит (його називають підзапитом) всередині речення WHERE. Підзапит повинен вибрати тільки одне поле, а тип даних цього поля повинен співпадати з тим значенням, з яким він порівнюється в предікаті.

EXISTS, ANY, ALL, SOME-спеціальні оператори, які завжди беруть підзапити у якості аргументів.

EXISTS-використовують, щоб вказати предикату на те, щоб виконувати виведення чи не виконувати виведення в підзапиті, при цьому EXISTS дає у якості результату значення ІСТИНА чи БРЕХНЯ. Наприклад, можливо вирішити, отримувати дані з таблиці успішності, якщо в ній присутні незадовільні оцінки. Це реалізується таким чином:

```
SELECT *
FROM USP
WHERE USP.OCENCA=2
AND EXISTS
(SELECT *
FROM USP
WHERE USP. OCENCA=2);
```

ANY, ALL та SOME, нагадують EXISTS, але відрізняються тим, що застосовуються разом з реляційними операторами, аналогічно IN в підзапитах. Наприклад:

```
SELECT *
FROM USP
WHERE OCENCA<>ALL
(SELECT OCENCA
FROM USP
WHERE UDATE=10/06/2009);
```

В цьому випадку підзапит вибере всі оцінки за 10.06.2009. Після цього основний запит виведе всі записи з оцінкою, яка не співпадає ні з одною з них.

Аналогічний результат можна отримати таким чином:

```
SELECT *
FROM USP
WHERE NOT OCENCA=ANY
(SELECT OCENCA
FROM USP
WHERE UDATE=10/06/2009);
```

Якщо запросити значення за допомогою команди ALL, не рівні набору значень, то це теж саме, що визнати факт відсутності цього значення у наборі.

DDL

Для зміни структури БД в SQL передбачено DDL (Data Definition Language) - мову визначення даних. За допомогою операторів DDL можливо наступне:

- Створити нову базу даних;

- Визначити структуру нової таблиці та створити цю таблицю;
- Видалити існуючу таблицю;
- Змінити визначення існуючої таблиці;
- Визначити представлення даних;
- Забезпечити умови безпеки БД;
- Створити індекси для доступа до таблиць;
- Керувати розміщенням даних на пристроях зберігання.

DDL базується на трьох командах SQL:

- CREATE-дозволяє визначити та створити об'єкт бази даних;
- DROP- застосовується для видалення існуючого об'єкту бази даних;
- ALTER -за допомогою якого можна змінити визначення об'єкта БД.

Команда CREATE TABLE-створює порожню таблицю з визначеним ім'ям та визначеним набором полів, які вказуються у певному порядку, а також визначаються типи полів. Наприклад:

```
CREATE TABLE STUDENTS
(SNUM INTEGER,
SFAM CHAR (20),
SIMA CHAR (10),
SOTCH CHAR (15));
```

Порядок розташування полів у таблиці визначається тим, в якій послідовності вони вказані в команді створення таблиці.

Команда ALTER TABLE-засіб для зміни визначення існуючої таблиці, якою можна додавати поля до існуючої таблиці(ADD), видаляти поля(DROP) чи змінювати їх розмір (ALTER). ALTER TABLE не діє, якщо необхідне перевизначення. Наприклад:

```
ALTER TABLE STUDENTS
ADD COURS INTEGER,
SPEC CHAR (10);
```

До таблиці STUDENTS додаються два поля для зберігання інформації про курс та спеціальність студента.

Для видалення таблиці використовують команду DROP TABLE. Таблица з інформацією не може видалятись.Таблиця видаляється, якщо вона пуста. Наприклад:

```
DROP TABLE STUDENTS;
```

Виконується видалення пустої таблиці STUDENTS.

Індексом називають впорядкований список полів чи груп полів в таблиці. Індекси-це корисний інструмент, який широко застосовується у всіх СКБД. Якщо створюється індекс у полі, БД запам'ятовує відповідний порядок всіх значень даного поля в області пам'яті. Таблиця, для якої створюється індекс повинна вже існувати і зберігати імена індексованих полів, при цьому ім'я індексів не може бути використано для чогось іншого в БД, а SQL сама вирішує коли він необхідний для роботи та користується ним автоматично. Наприклад команда для створення індексу по полю, яке зберігає прізвище студента:

```
CREATE INDEX SFAMIDX ON STUDENTS (SFAM);
```

Для створення унікальних індексів використовується ключове слово UNIQUE у команді CREATE INDEX. Фактично такий індекс буде первинним ключем таблиці.

Створити унікальні індекси можна так:

```
CREATE UNIQUE INDEX
SNAMIDX ON STUDENTS (SNAM);
```

Ця команда не виконається, якщо в полі SNAM трапляться неунікальні значення.

Для видалення створеного індексу за прізвищем студента, можна скористатись командою:

```
DROP INDEX SFAMIDX
ON STUDENTS;
```

Обмеження даних – це частина визначень таблиці, яка обмежує значення, які допускаються до введення в поля таблиці. Обмеження можна вказувати, коли створюється чи змінюється таблиця. Існують два основних види обмежень. Обмеження поля та обмеження таблиці. Обмеження поля ставлять у кінець фрагмента команди, яка оголошує його ім'я після типу даних. Обмеження таблиці ставлять у кінець оголошення імені таблиці. NOT NULL-оберігає поле від порожніх значень.

Є можливість встановити унікальність в якості обмеження стовпця за допомогою ключового слова UNIQUE. А за допомогою обмеження PRIMARE KEY можливо обмежувати таблицю чи окремі стовпці таблиці. Синтаксис та визначення його унікальності такі ж які UNIQUE- первинні ключі не допускають NULL значень, тому перед такими обмеженнями необхідно оголосити NOT NULL. Наприклад:

```
CREATE TABLE YSP
(UNUM INTEGER NOT NULL PRIMARY KEY,
OCENKA INTEGER,
UDATE DATE,
SNUM INTEGER NOT NULL,
PNUM INTEGER NOT NULL,
UNIQUE (SNUM, PNUM));
```

Як видно, краще накладати обмеження на поле, яке створює унікальний ідентифікатор рядка, і зберегти обмеження UNIQUE для поля чи групи полів, які повинні бути унікальними логічно, а не застосовуватись для ідентифікації рядків.

Контрольні питання до лабораторної роботи 2:

1. **Які запити називають багатотабличними запитами?**
2. **Що означає UNION в інструкції SELECT?**
3. **Які функції не можна застосовувати в інструкції SELECT запиту на об'єднання?**
4. **Чи дозволяється створювати запити, які об'єднують більше двох таблиць?**
5. **Який запит називають підзапитом?**
6. **Які умови накладаються на значення яке отримується у підзапиті?**
7. **Як відокремлюються одна від одної таблиці в багатотабличних запитах, які представлені у вигляді списку в інструкції FROM?**
8. **Які дії виконуються спочатку для запита, який використовує виведення внутрішнього запиту?**
9. **Які спеціальні оператори завжди беруть підзапити у якості аргументів?**
10. **В яких випадках використовують оператор EXISTS?**
11. **Чим відрізняються оператори ANY, ALL та SOME від EXISTS?**
12. **Які можливості DDL?**
13. **На яких командах базується DDL?**
14. **За допомогою якої команди можна створити таблицю?**
15. **За допомогою якої команди можна видалити таблицю?**
16. **За допомогою якої команди можна змінити визначення об'єкта БД?**
17. **Що означає команда CREATE TABLE?**
18. **Що означає команда ALTER TABLE?**
19. **Що означає команда DROP TABLE?**
20. **Які дії над таблицею треба виконати перед тим як видалити її?**
21. **Що називають індексом?**
22. **Що означає обмеження даних?**
23. **Коли не можна створити унікальний індекс в таблиці з даними?**
24. **Що означає обмеження NOT NULL у кінці оголошення імені таблиці?**

Лабораторна робота № 3

Тема: Проектування і використання баз даних. Автоматизація роботи з базою даних за допомогою макросів.

Мета: Навчитися автоматизувати роботу з БД мовою макросів.

Зміст роботи за варіантом індивідуального завдання:

1. Побудуйте простий макрос, який відкриває запит, таблицю, форму.
2. Побудуйте макрос, який відкриває форму у випадку пустих значень у таблиці.
3. Об'єднайте макроси у макрогрупу.
4. Створіть на формі кнопку, при натисненні якої викликався б один з макросів із макрогрупи.
5. Створіть на формі кнопку, при натисненні якої викликавсяб макрос з командою попереднього перегляду звіту, якщо сьогоні 29 число, інакше відкривалась би форма для введення даних.
6. Автоматизуйте виведення на друк за допомогою макросу.
7. За допомогою макросів автоматизуйте операції, що повинні завжди виконуватися в момент завантаження бази.

Теоретичні відомості:

Мова макросів - це засіб автоматизації БД, дозволяє перетворити виснажливе повторення послідовностей операції в просту процедуру, що складається лише в натисканні спеціальної кнопки або комбінації клавіш, або в активізації меню.

Для відкриття вікна макросів вибираєте ***Создание,*** в секції Другие, Макросо включає чотири стовпчики (рисунок 1): ***Имя макроса, Условие, Макрокоманда*** і ***Примечание***.

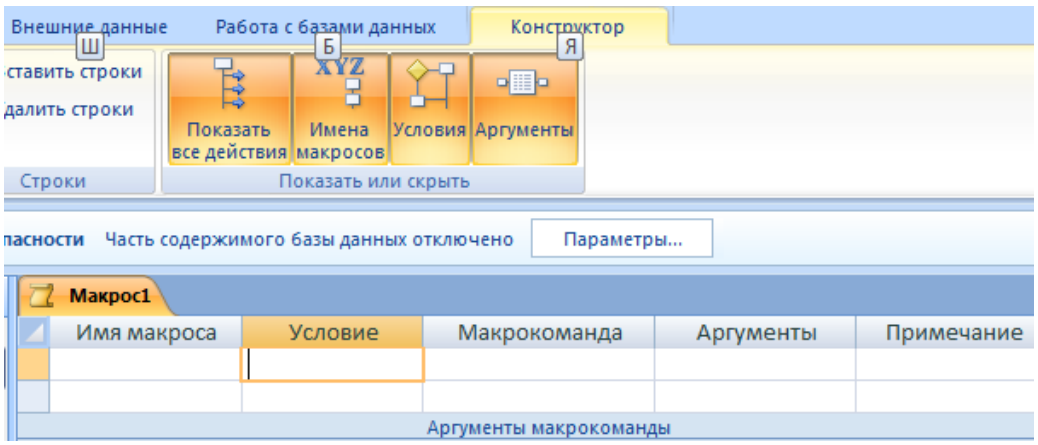

Рисунок 1 –Нове вікно макросів

При створенні нового макроса по замовчуванню відображаються тільки стовпчики ***Макрокоманда і Примечание***. Показ інших установлюється за допомогою вибору відповідних опцій на вкладці Конструктор.

У стовпчику ***Имя макроса*** вказується ім'я макроса, що слід задавати, якщо вікно містить декілька макросів. У стовпчику ***Условие*** використовують умовний вираз. Це такий вираз, значення якого Microsoft Access перевіряє та порівнює з вказаним значенням, наприклад, в інструкціях If...Then та Select Case. Якщо умова порівняння задовільняється, то виконується одна або декілька операцій. Якщо умова не

задовільняється, то включені в умовну конструкцію операції пропускаються, та виконується перехід на наступну інструкцію.

У стовпчику ***Макрокоманда*** перераховуються дії (макрокоманди), що підлягають виконанню у потрібній послідовності (необхідно вибрати з списку потрібну).

Зручно розробляти макроси для автоматизації нескладних процесів, таких як відкриття і закриття декількох форм або звітів, виведення на екран або друк декількох документів і т.д.

Виконання кожної макрокоманди залежить від значень її аргументів. Всі аргументи вводяться в спеціально відведені для цього поля, розташовані в нижній частині вікна макросу. Аргументи рекомендується задавати в тому порядку, у якому вони розташовані в нижній частині вікна макросу. Розробка макросу починається з запровадження макрокоманд відкриття об'єктів, перетаскуванням відповідних об'єктів із вікна бази даних в комірки вікна конструктора макросу. При цьому Access автоматично розпізнає, про який об'єкт йде мова, і вибирає відповідну макрокоманду: OpenForm для форми або OpenTable для таблиці. Ім'я переміщеного об'єкта з'являється в області аргументів макрокоманди в якості значення параметра ***Имя Макроса***. Важливо, щоб до моменту виконання макросу, що відчиняється об'єкт вже існував.

Такий аргумент макрокоманди відкриття об'єктів, введений у поле ***Вид***, визначає режим відображення об'єкта на екрані. Припустимі значення цього аргументу відповідають опціям із меню ***Вид***.

За допомогою макрокоманди ***ВыполнитьКоманду*** можна задати виконання більшості команд із меню. Ім'я виконуваної команди вказується в якості аргументу в полі ***Команда***.

Редагування макросу здійснюється в режимі конструктора.

Об'єднання макросів у макрогрупу

Для роботи з одним об'єктом бази даних можуть знадобитися десятки макросів, кожний із який автоматизує деякий процес. Такі макроси можна розробити окремо і запускати по черзі. Проте всі макроси, призначені для опрацювання одного об'єкта, доцільно об'єднати в групу, привласнивши їм унікальні в межах групи імена.

Для створення групи макросів необхідно (рисунок 2):

- Відчиніть вікно конструктора макросу, у який заплановано помістити макроси, що групуються.
- Відобразити на екрані стовпчик ***Имя макроса***.
- У перший осередок стовпчика ***Имя макроса*** введіть ім'я першого макросу.
- У комірку стовпчика ***Макрокоманда*** введіть макрокоманди, що складають тіло проектованого макросу.
- У наступному рядку в поле ***Имя макроса*** введіть ім'я наступного макросу.
- Напишіть інші макроси і збережіть макрогрупу.

Рисунок 2

Виклик макросу з макрогрупи здійснюється шляхом вказівки точних імен макрогрупи і макросу в групі. Ці імена розділяються крапкою. У створену макрогрупу в будь-який момент можна додати нові макроси.

Зв'язування макросів із подіями

У базі даних зберігається інформація про стан її об'єктів. Будь-яка зміна стану форми або звіту називається подією. Кожний із цих об'єктів має свій набір подій. З подіями зручно зв'язувати макроси.

Для того щоб зв'язати макрос із відкриттям бази даних виконайте такі дії:

- Виберіть макрос у вікні бази даних.
- Змініть ім’я макроса на - Autoexec (макрос із таким ім'ям автоматично виконується при завантаженні бази даних).

Тепер макрос пов'язаний із відкриттям бази даних. Таким засобом автоматизуються операції, що повинні завжди виконуватися в момент завантаження бази. При кожному відкритті бази даних Access перевіряє, чи є присутнім в ній макрос з ім'ям Autoexec, і, якщо знаходить, виконує його.

Автоматизація виводу на друк

Друк може бути автоматизований за допомогою макросів. Для виводу на друк в залежності від версії передбачені різні макрокоманди. Якщо необхідно роздрукувати звіт:

- Вибираємо макрокоманду ОткрытьОтчет, вказуємо який сааме звіт.
- Вибираємо макрокоманду Выполнить команду, вказуємо команду Печать, якщо необхідно, можливо вибрати спочатку попередній перегляд з списку команд.

Приклад демонструє використання макросу для виведення звіту на друк. Макрос перевіряє, чи відповідає поточне число місяця, якщо так, то друкує звіт (рисунок 3).

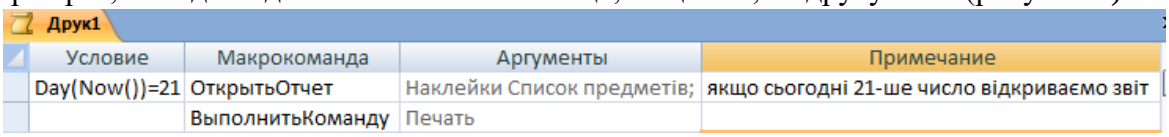

Рисунок 3

Покроковий режим

Виконання макросу в покроковому режимі дозволяє прослідкувати передачу керування та результатів виконання кожної макрокоманди. Це полегшує пошук макрокоманди, яка викликає помилку або повертає не хибні результати.

1. Відкрийте макрос у режимі конструктора.
2. Натисніть кнопку **По шагам** на панелі інструментів.
3. Натисніть кнопку **Запуск**.
4. Натисніть кнопку **Шаг** для виконання макрокоманди, ім'я якої виведено у діалоговому вікні **Пошаговое исполнение макроса**.
5. Натисніть кнопку **Остановить все макросы**, щоб припинити виконання макросу та зачинити вікно діалогу.
6. Натисніть кнопку **Продолжить**, щоб відключити режим покрокового виконання та виконати частину макросу, що залишилась.

Контрольні питання до лабораторної роботи 3:

1. **Що представляє собою об'єкт Макросы?**
2. **Які стовпчики включає вікно макросів у режимі конструктора?**
3. **Які дані вказуємо у стовпчику Условие вікна макросів?**
4. **Як налаштувати дії кожної з макрокоманд у макросах? У якому місці вони вказуються?**
5. **Які дії можна автоматизувати за допомогою макросів?**
6. **Що представляє собою макрогрупа?**
7. **Чи можливо запустити на виконання один з макросів із макрогрупи?**
8. **Як пов'язати макрос із відкриттям бази даних?**
9. **В яких випадках застосовують макроси?**
10. **Що дозволяє покроковий режим виконання макросу?**

Лабораторна робота № 4

Тема: Проектування і використання баз даних. Розробка додатків користувачів за допомогою VBA.

Мета: Навчитися розробляти додатки на мові Visual Basic.

Зміст роботи за варіантом індивідуального завдання:

1. Наведіть фрагмент модуля форм вашої БД, який має умовні оператори.
2. Наведіть фрагмент модуля форм вашої БД, який має оператори циклу.
3. Наведіть фрагмент модуля форм вашої БД, який має оператор CASE.
4. Створіть підпрограму, яка б після натискання кнопки виводила напис «Введіть текст». При введенні «Вихід» – відбувається закриття форми, якщо інше – виводиться напис «Роботу продовжено».
5. Створіть процедуру-функцію, за допомогою якої обчислюються значення, що використовуються за замовчуванням для деякого поля вашої БД.

Теоретичні відомості:

Access дозволяє автоматизувати опрацювання баз даних і шляхом програмування на мові Visual Basic for Application.

Для збереження коду мови VBA застосовуються модулі - самостійні об'єкти, кожен із яких містить одну або декілька процедур. У Access використовують модулі таких типів: стандартні модулі, модулі форм і звітів. На відміну від стандартного модуля, що створюється таким же чином, як і будь-який інший об'єкт баз даних, і може виконувати практично будь-які обчислення, модулі форм і звітів розробляються для опрацювання подій, пов'язаних з елементами форми або звіту.

Кожен модуль складається з області опису й однієї або декількох процедур. Процедура являє собою послідовність операторів, що часто називаються програмними кодами. Процедури, що входять до модулю, об'єднані загальною областю опису. У ній описуються дані й об'єкти, які є звичайними для процедур модуля. Ієрархія зазначених об'єктів така:

- база даних;
- модуль;
- область опису;
- процедура;
- код;
- оператор.

Процедури поділяються на дві категорії: процедури-підпрограми (підпрограми) і процедури-функції (функції).

Процедури-підпрограми активізуються при зверненні до них по імені, внаслідок чого виконується визначена послідовність операторів (інструкцій). Підпрограми використовують, наприклад, для завдання властивості форми або для заповнення списку значеннями, отриманими в результаті обчислень.

Процедура-функція після виконання повертає деяке значення, що можна застосовувати в операторах і виразах у якості змінної, наприклад, функції можуть повертати значення, що використовується за замовчуванням для деякого поля або

обчислювати складний критерій у рамках запиту. У модуль можна включати будь-яку кількість функцій і підпрограм.

Процедура складається з послідовності операторів, у яких застосовуються вбудовані в Access функції, методи і властивості. Для звернення до перерахованих об'єктів, а також для позначення операторів використовуються ключові слова, що записуються з прописної букви (наприклад, Function).

Теоретично кожна процедура-підпрограма може бути викликана з будь-якого модуля, а функція - із таких об'єктів, як форма, запит і звіт. Поряд із загальнодоступними процедурами (Public), якими є всі процедури за замовчуванням, існують локальні, або особисті, процедури (Private), доступні тільки в тому модулі, у якому вони описані.

Для передачі значень операторів у процедури, що викликаються служать аргументи. За допомогою аргументів ведеться контроль за виконанням процедури, установлюється засіб одержання результату, визначаються параметри обчислення і т.д.

Елементи вікна модуля

Модулі створюються в спеціальному вікні. У процесі упорядкування розроблювач використовує текстовий редактор, призначений для запровадження програмного модуля.

Модули створюють так: на вкладці **Создание**, вибрать **Другие**, **Модули**. Вікно модуля з'являється при відкритті існуючого модуля або при створенні нового.

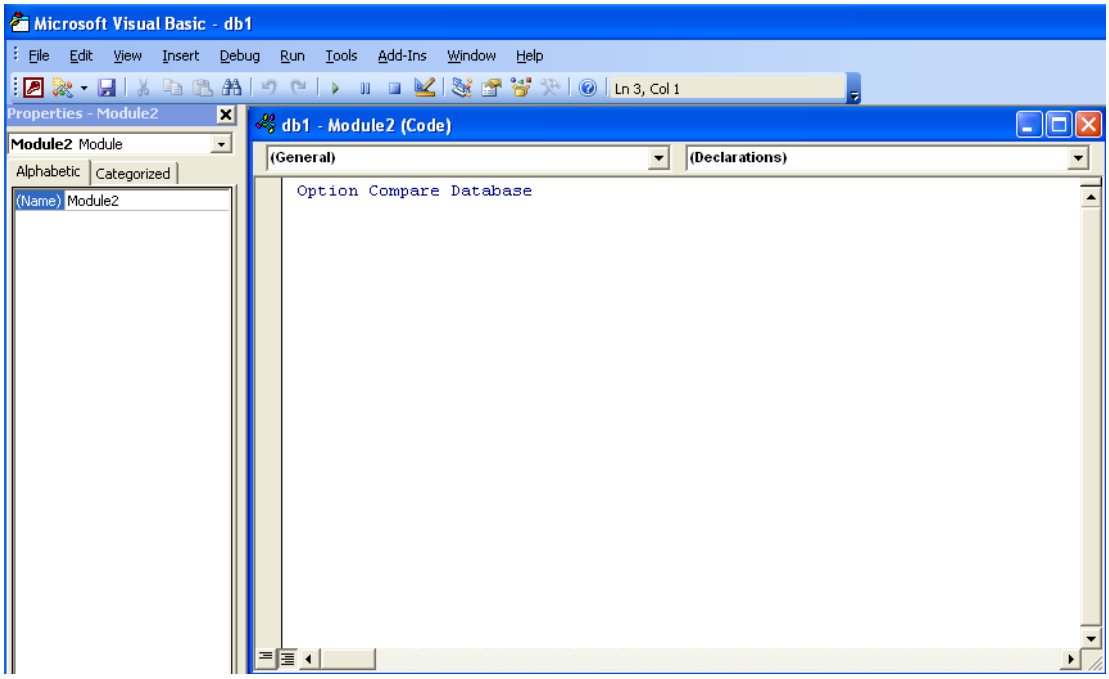

Рисунок 1

Розділ описів

Відразу після відкриття вікно нового модуля містить тільки розділ опису. У цьому розділі здійснюється встановлення змінних і констант, що використовуються у підпрограмах і функціях модуля. За замовчуванням в розділі описів нового модуля з'являються такі оператори (рис.1):

Option Compare Database

При створенні нового модуля Access добавляє в розділ опису оператор Option Compare Database, за допомогою якого встановлюється режим порівняння текстових даних у модулі. У Access передбачені три режими порівняння текстових даних у

модулі. Якщо в розділі описів відсутній відповідний оператор, за замовчуванням активізується режим Binary.

Створення процедури

Для створення процедури (підпрограми або функції) у Access призначена команда **Procedure** із меню **Insert**. У результаті її активізації відчиняється діалогове вікно **Insert Procedure**, що служить для вибору типу процедури (**Sub** (підпрограма) або **Function** (функція)) і привласненні їй імені (поле **Name**). Задають область її використання за допомогою таких перемикачів:

-**Public** (Загальнодоступна) - доступна для всіх процедур у всіх модулях.

-**Private** (Особиста) - процедура доступна для інших процедур тільки в тому модулі, у якому вона оголошена.

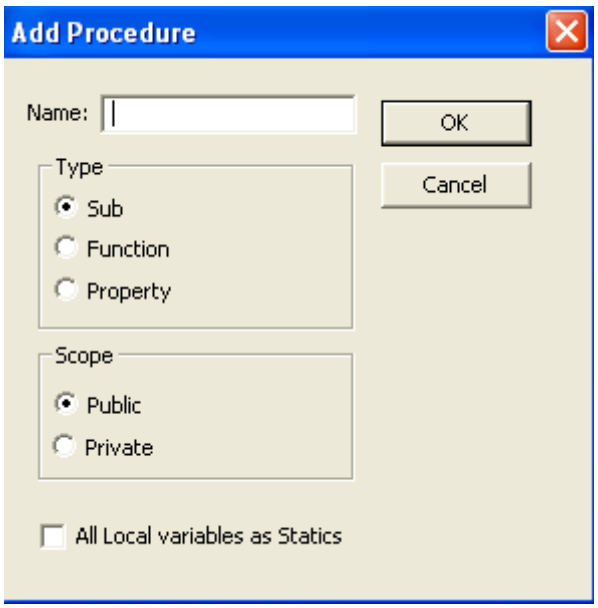

Рисунок 2

Аргументи процедури

Завдяки аргументам користувач має можливість керувати виконанням процедури.

Виклик процедури, що має аргументи, повинний супроводжуватися завданням значень для всіх оголошених аргументів. Ці значення перераховуються в тому ж порядку, що й аргументи при оголошенні, і розділяються комами. Якщо один з аргументів буде пропущений, то компілятор повідомить про помилку.

У користувацьких функціях або модулях у якості аргументів можуть застосовуватися вирази. Це припускається в багатьох поширених вбудованих функціях.

Змінні та оператор присвоювання в VBA

Присвоєння значення змінної здійснюється за допомогою оператора присвоювання, в якому ліворуч вказується ім'я змінної, а праворуч - значення, що привласнюється або вираз. Рекомендується описувати всі змінні явно, що дозволить уникнути помилок, пов'язаних із перетворенням типів даних. Явний опис змінних виконується за допомогою оператора **Dim**, після якого вказується ім'я змінної і її тип:

```
Dim i As Integer
Dim P As Double
Dim Str As String
i = 5
P = 3,14
Str = “Something”
```

Умовні оператори

У мові VBA, як і в інших мовах програмування основними елементами, що керують ходом виконання процедури, є умовні оператори. Найбільш простий із них - оператор IF … THEN:

```
IF i = 5 THEN Day = “Friday”
END IF
```

У разі потреби зробити дві різноманітні дії слід скористатися повною формою оператора IF:

```
IF i = 5 THEN Day = “Friday”
ELSE
Day = “Saturday”
END IF
```

Далеко не у всіх ситуаціях можливі два варіанти рішення. З огляду на це, VBA надає в розпорядження користувачів оператор SELECT CASE, призначений для вибору одного з множини варіантів:

```
    SELECT CASE i
    CASE 1
    Day = “Monday”
    CASE 2
    Day = “Tuesday”
… … … … …
    CASE 7
    Day = “Sunday”
    END SELECT
```

Оператори циклу

Найпростішим прикладом використання циклічної конструкції є цикл із лічильником:

```
    Dim Лічильник As Integer
    For  Лічильник = 1 To 10
    Print Лічильник
    Next Лічильник
```

Ще один різновид циклу - While-цикл. Умова виконання команд усередині такого циклу визначається деяким умовним оператором.

```
    Dim Лічильник As Integer
    Лічильник = 1
    Do While Лічильник<>10
    Print Лічильник
    Loop Лічильник
```

Приклад № 1:

Написати процедуру, яка б після натискання кнопки виводила напис «Введіть число». Якщо користувач ввів 0 – відбувається закриття форми, якщо інше – виводиться напис «Невірне число».

```
Private Sub Кнопка0_Click()
Dim i As Integer
i = InputBox("Введіть число")
If i = 0 Then
```

```
DoCmd.Close acForm, "форма1"
Else
InputBox ("Невірне число")
End If
End Sub
```

Приклад № 2:

У даному прикладі розглядається процедура опрацювання події, пов'язаної з натисканням кнопки «Вихід» форми, що відкривала декілька таблиць для редагування.

```
Private Sub ЗакриттяФорми1_Click()
‘В разі помилки перехід на мітку
On Error GoTo Err_ЗакриттяФорми1_Click

‘Закриваємо форму
   DoCmd.Close acForm, "Форма1"
 ‘Зберігаємо зміни, що були зроблені у відкритих
  ‘ таблицях
   DoCmd.Save acTable, "Табель виходу на роботу"
   DoCmd.Save acTable, "Утримано"
   DoCmd.Save acTable, "Начислено"
  ‘Закриваємо таблиці
   DoCmd.Close acTable, "Табель виходу на роботу"
   DoCmd.Close acTable, "Утримано"
   DoCmd.Close acTable, "Начислено"
   ‘Відкриваємо форму “Заставка”
   DoCmd.OpenForm "Заставка"
‘В разі помилки вихід з підпрограми
Exit_ЗакриттяФорми1_Click:
   Exit Sub
‘Мітка, на яку здійснюється перехід в разі помилки
Err_ЗакриттяФорми1_Click:
‘Видається повідомлення про помилку
   MsgBox Err.Description
   Resume Exit_ЗакриттяФорми1_Click
End Sub
```

Приклад № 3:

В наступному прикладі функція повертає текстове значення поточного місяця.

```
Option Compare Database
Option Explicit

Public Function DoMonth() As String
Dim m
‘Змінній m присвоюється цифрове значення поточного ‘місяця
m = Month(Now())
‘За значенням змінної m функцій присвоюється тектове ‘значення поточного місяця
Select Case m
```

```
Case 1
DoMonth = "Січень"
Case 2
DoMonth = "Лютий"
Case 3
DoMonth = "Березень"
Case 4
DoMonth = "Квітень"
Case 5
DoMonth = "Травень"
Case 6
DoMonth = "Червень"
Case 7
DoMonth = "Липень"
Case 8
DoMonth = "Серпень"
Case 9
DoMonth = "Вересень"
Case 10
DoMonth = "Жовтень"
Case 11
DoMonth = "Листопад"

Case 12
DoMonth = "Грудень"

End Select
End Function
```

Приклад № 4:

В наступному прикладі створюється нова таблиця, поля таблиці та визначається ключ таблиці.

```
Function CreateNewTable()
Dim dbs As Database
Dim tdfNew As TableDef
Dim Indx As Index
‘Змінна dbs вказує на поточну базу даних
Set dbs = CurrentDb()
‘Змінна tdfNew вказуватиме на нову створену таблицю ‘“Підсумкові дані”
Set tdfNew = dbs.CreateTableDef("Підсумкові дані")
‘Створюються поля нової ‘таблиці та задається їх формат

    With tdfNew
      .Fields.Append .CreateField("Табельний номер", dbInteger)
      .Fields.Append .CreateField("Кількість годин", dbInteger)
      .Fields.Append .CreateField("Основна зарплатня", dbDouble)
       .Fields.Append .CreateField("Сума відпустки", dbDouble)
      .Fields.Append.CreateField("Доплата за інтенсивність", dbDouble)
      .Fields.Append.CreateField("Доплата за шкідливість", dbDouble)
      .Fields.Append .CreateField("Нічні", dbDouble)
```

```
        .Fields.Append .CreateField("Премія", dbDouble)
        .Fields.Append.CreateField("Сума по хворобі", dbDouble)
        .Fields.Append.CreateField("Всього начислено", dbDouble)
        .Fields.Append.CreateField("Сума до пенсійного фонду", dbDouble)
        .Fields.Append.CreateField("Сума прибуткового податку", dbDouble)
        .Fields.Append.CreateField("Сума аліментів", dbDouble)
        .Fields.Append.CreateField("Сума по виконавчим листам", dbDouble)
        .Fields.Append.CreateField("Фонд безробіття", dbDouble)
        .Fields.Append.CreateField("Сума до фонду Чорнобиля", dbDouble)
        .Fields.Append.CreateField("Всього утримано", dbDouble)

        .Fields.Append.CreateField("Сума до виплати", dbDouble)

    End With
‘Нова таблиця “Підсумкові дані ” додається до
‘сімейства TableDefs
    dbs.TableDefs.Append tdfNew

    With tdfNew
    'створення унікального індексу (ключа таблиці)
    Set Indx = .CreateIndex("SomeIndex")
’Створюється нове індексне поле
    Indx.Fields.Append Indx.CreateField("Табельний номер", dbInteger)
‘Встановлюється ознака унікальності поля
    Indx.Primary = True
‘Додається нове індексне поле до сімейства Indexes
    .Indexes.Append Indx
    End With

End Function
```

Приклад № 5:

В наступному прикладі робляться підрахунки. Є три таблиці, дані з яких підраховуються та заносяться у таблицю “Підсумкові дані”. Для доступу до записів таблиць використовується сімейство RecordSet.

```
Option Compare Database
Option Explicit

Public Function DoMath()
Dim dbs As Database
Dim Hours  As Integer
Dim WholePay, PayMin, PayTax, PensTax As Double
Dim i As Long
‘Змінна dbs вказує на поточну базу даних
 Set dbs = CurrentDb()
‘Далі відкриваються записи 4-ох таблиць
Dim Record1 As Recordset
    Set Record1 = _
    dbs.OpenRecordset("Табель виходу на роботу", dbOpenTable, dbReadOnly)
Dim Record2 As Recordset
    Set Record2 = _
```

```
   dbs.OpenRecordset("Підсумкові дані", dbOpenTable)
Dim Record3 As Recordset
    Set Record3 = _
   dbs.OpenRecordset("Начислено", dbOpenTable)
 Dim Record4 As Recordset
    Set Record4 = _
   dbs.OpenRecordset("Утримано", dbOpenTable)
‘Переходимо на перший запис кожної таблиці
   Record1.MoveFirst
   Record3.MoveFirst
   Record4.MoveFirst
   i = 1
‘ Цикл поки не  буде зроблено підрахунки по
‘ всіх записах таблиці
Do While i <= Record1.RecordCount
   Hours = 0
Hours = Record1![1-й день]
Hours = Hours + Record1![2-й день]
Hours = Hours + Record1![3-й день]
……..
Hours = Hours + Record1![30-й день]
Hours = Hours + Record1![31-й день]
‘Далі йдуть підрахунки
'Підрахунок загальної зарплатні
WholePay = Hours * Record1![Оклад] + Record3![Відпустка] + (Record3![Доплата за інтенсивність] * Hours * Record1![Оклад]) + Record3![Премія] * Hours * Record1![Оклад]
'Загальна сума утримань
PayMin = WholePay * (Record4![Пенсійний фонд] + Record4![Прибутковий податок] + Record4![Аліменти] + Record4![По виконавчим листам] + Record4![Фонд безробіття] + Record4![Фонд Чорнобиля] + Record4![Профсоюзний взнос])
'Прибутковий податок та до пенсійного фонду
Select Case WholePay
Case 17 To 85
PayTax = (WholePay * 10) / 100
Case 86 To 170
PayTax = (WholePay * 15) / 100
Case 170 To 1000
PayTax = (WholePay * 20) / 100
End Select
Select Case WholePay
Case Is <= 150
PensTax = (WholePay * 1) / 100
Case Is > 150
PensTax = (WholePay * 2) / 100
Case Is >= 3000
PensTax = (WholePay * 32) / 100
End Select
‘До таблиці “Підсумкові дані” заносимо дані, що були ‘підраховані
With Record2
.AddNew
```

```
![Табельний номер] = Record1![Табельний номер]
![Кількість годин] = Hours
![Основна зарплатня] = Hours * Record1![Оклад]
![Сума відпустки] = Record3![Відпустка]
![Сума за інтенсивність] = Record3![Доплата за інтенсивність] * Hours * Record1![Оклад]
![Сума за шкідливість] = Record3![Доплата за шкідливість] * Hours * Record1![Оклад]
![Сума нічних] = Record3![Нічні] * Hours * Record1![Оклад]
![Сума премії] = Record3![Премія] * Hours * Record1![Оклад]
'![Сума по хворобі] =
![Всього начислено] = WholePay
'Утримання
![Сума до пенсійного фонду] = PensTax
![Сума прибуткового податку] = PayTax
![Сума аліментів] = Record4![Аліменти] * WholePay
![Сума по виконавчим листам] = Record4![По виконавчим листам] * WholePay
![Сума до фонду безробіття] = Record4![Фонд безробіття] * WholePay
![Сума до фонду Чорнобиля] = Record4![Фонд Чорнобиля] * WholePay
![Сума профвзносу] = Record4![Профсоюзний взнос] * WholePay
![Всього утримано] = PayMin
![Сума до виплати] = WholePay - PayMin
.Update
End With
Hours = 0
WholePay = 1
PayMin = 1
PayTax = 1
PensTax = 1
i = i + 1
‘Перехід до наступних записів таблиць
Record1.Move 1
Record3.Move 1
Record4.Move 1
Loop
End Function
```

Контрольні питання до лабораторної роботи 4:

1. **Що представляє собою об’єкт Модулі?**
2. **Яких типів модулі використовують у Access?**
3. **З яких областей складається кожен модуль?**
4. **Що являє собою процедура?**
5. **Які бувають процедури?**
6. **Для чого служать аргументи?**
7. **Який оператор об’яви змінних?**
8. **Перелічіть основні оператори VBA, які використовували в роботі.**
9. **Чим відрізняється процедура підпрограми від процедури функції?**
10. **Що означає специфікатор Public?**
11. **Що означає специфікатор Private?**

Лабораторна робота №5

Тема: Зберігання інформації у базах даних. Захист бази даних, OLE-об’єкти у БД.

Мета: Ознайомитися з захистом баз даних. Навчитися працювати з OLE-об’єктами у БД.

Зміст роботи за варіантом індивідуального завдання:

1. Створити надійне розташування для файлів БД.
2. Упакуйте та підпишіть БД.
3. ЗашифруйтеБД за допомогою пароля.
4. Видаліть пароль.
5. Побудуйте **.accde** -файл.
6. Захистіть модулі Visual Basic за допомогою пароля.
7. Використайте об'єкт OLE в формі БД.
8. Зробіть так, щоб відкривалася форма з вашої БД з одним фоновим малюнком для режиму введення даних і з іншим для режиму перегляду.

Теоретичні відомості:

Центр безпеки та конфіденційності. Центр безпеки та конфіденційності є діалоговим вікном, де можна задати та змінити всі параметри безпеки в Access. Центром безпеки та конфіденційності користуються для створення та змінення надійних розташувань і встановлення параметрів безпеки для Office Access 2007. Ці настройки впливають на поведінку нових і наявних баз даних, коли вони відкриваються у цьому екземплярі Access. У Центрі безпеки та конфіденційності також можна перевіряти компоненти бази даних і визначати, чи є база даних безпечною для відкриття або Центр безпеки та конфіденційності має вимкнути цю базу даних, а рішення про ввімкнення буде приймати користувач.

Вибрати надійне розташування

1. Натисніть кнопку **Microsoft Office**  , а потім – кнопку **Параметри Access** (не відкривати БД). З’явиться діалогове вікно **Параметри Access**.
2. Виберіть пункт **Центр безпеки та конфіденційності** і у розділі **Центр безпеки та конфіденційності Microsoft Office Access** виберіть пункт **Настройки центру безпеки та конфіденційності**.
3. Виберіть пункт **Надійні розташування**, а потім виконайте одну з таких дій:

a. Занотуйте шлях до одного або кількох надійних розташувань.

b. Створіть нове надійне розташування. Для цього виберіть команду **Додати нове розташування**, а потім виберіть потрібні параметри в діалоговому вікні **Надійне розташування Microsoft Office**.

Упакувати та підписати

1. Відкрийте базу даних, яку потрібно упакувати та підписати.
2. Натисніть **кнопку Microsoft Office**  , виберіть розділ **Опублікувати**, а потім виберіть пункт **Упакувати й підписати**. З’явиться діалогове вікно **Вибір сертифіката**.

3. Виберіть цифровий сертифікат і натисніть кнопку **ОК**. З'явиться діалогове вікно **Створити підписаний пакет Microsoft Office Access**.
4. У списку **Зберегти в** виберіть розташування для підписаного пакета бази даних.
5. Введіть ім'я пакета в полі **Ім'я файлу** та натисніть кнопку **Створити**.

Програма Access створює файл із розширенням .accdc і розміщує його у вибраному розташуванні.

Використання пароля бази даних для шифрування бази даних Office Access 2007

Засіб шифрування у Office Access 2007 поєднує та покращує два старіших засоби — кодування та паролі баз даних. Якщо для кодування бази даних застосовується пароль бази даних, то всі дані не можна прочитати за допомогою інших знарядь, й інші користувачі змушені вводити пароль, щоб користуватися базою даних. Шифрування, що застосовується у Office Access 2007, використовує кращий алгоритм, ніж у попередніх версіях Access.

Шифрування за допомогою пароля бази даних

1. Відкрийте БД, яку потрібно зашифрувати, у монопольному режимі.

Відкриття бази даних у монопольному режимі

- Натисніть кнопку **Microsoft Office** і виберіть пункт **Відкрити**.
- У діалоговому вікні **Відкрити** знайдіть файл, який потрібно відкрити, і виділіть його.
- Клацніть стрілку, розташовану поруч із кнопкою **Відкрити**, а потім виберіть команду **Монопольний доступ (рис. 1)**.

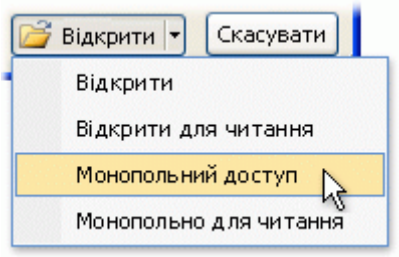

Рисунок 1

2. На вкладці **Знаряддя бази даних** у групі **Знаряддя бази даних** натисніть кнопку **шифрувати паролем**.

Відображається діалогове вікно **Установлення пароля бази даних**.

3. Введіть пароль у полі **Пароль** і введіть його ще раз у полі **Перевірка** (великі та малі букви, цифри та символи, 8 або більше символів).
4. Натисніть кнопку **ОК**.

Розшифрування та відкриття бази даних

1. Відкрийте зашифровану базу даних, як будь-яку іншу. З'явиться діалогове вікно **Потрібний пароль**.
2. Введіть пароль у полі **Введіть пароль бази даних**, а потім натисніть кнопку **ОК**.

Видалення пароля

- На вкладці **Знаряддя бази даних** у групі **Знаряддя бази даних** натисніть кнопку **Розшифрувати базу даних**.

З'явиться діалогове вікно **Видалення пароля бази даних**.

- Введіть пароль у полі **Пароль** і натисніть кнопку **ОК**.

Якщо база даних містить Visual Basic для додатків (VBA), можна приховати код, зберігши свій Microsoft Office Access базу даних в форматі .accde. Збереження бази даних як файл .accde компілює всі модулі коду VBA, видаляє всі редагування вихідного

коду і стискає кінцеву базу даних. Код VBA зберігає функціональність, але не можна переглянути або змінити код. Як правило, бази даних будуть працювати як завжди, можна як і раніше оновлення даних і запуск звітів.

Створення .accde файлу

- На вкладці **Знаряддя бази даних** у групі **Знаряддя бази даних** натисніть кнопку Создать ACCDE. З'явиться діалогове вікно.
- Підтвердіть збереження файлу і місце розташування.

Встановлення пароля на модулі VB

- Відкрити вікно редактору коду. В меню **Tools** редактора Microsoft Visual Basic виберіть команду **<имя базы данных Access или проекта Access> Project Properties.** На вкладці **Protection** встановіть флажок **Lock project for viewing**.
- Введіть пароль в поле **Password**.

Об’єкти OLE

Механізм зв’язування і вкорінювання об’єктів, або коротко технологія OLE (скорочення від англ. Object Linking and Embedding – зв’язування і вкорінювання об’єктів), є удосконалюванням засобів DDE. Вкорінювання означає, що програма, яка використовує вкорінений, тобто створений іншою програмою об’єкт, сприймає його як свій власний. При цьому є можливість редагувати “на місці”: подвійне клацання мишею на вкоріненому об’єкті запускає відповідну програму. Механізм зв’язування додає зв’язок укоріненого об’єкта з оригіналом, отже всі зміни оригіналу відбиваються на його вкоріненій копії.

Розглянемо властивості OLE об’єктів.

Класс (Class) дозволяє вказати або визначити ім'я класу впровадженого об'єкту OLE.

Значенням властивості Клас (Class) є строковий вираз, що задається користувачем або автоматично підставляється в Microsoft Access при створенні або вставленні об'єкту OLE.

Користувач задає значення властивості Клас (Class) в вікні властивостей елементу керування, в макросі або в програмі Visual Basic.

Ім'я класу визначає тип об'єкту OLE. Наприклад, Microsoft Excel підтримує декілька типів об'єктів OLE, в тому числі електронні таблиці і діаграми. Іменами класів для даних об'єктів є, відповідно, “Excel. Sheet” і “Excel. Chart”. Якщо користувач створює об'єкт OLE в режимі конструктора за допомогою команди **Специальная вставка** з меню **Правка** або команди **Объект** з меню **Вставка**, то ім'я класу нового об'єкту автоматично підставляється в вікно властивостей.

Значення властивості **Класс (Class)** оновлюється при копіюванні об'єкту з буферу обміну. Наприклад, при вставці діаграми Microsoft Excel з буферу обміну в об'єкт OLE, який до цього містив електронну таблицю Microsoft Excel, значення властивості Клас (Class) зміниться з “Excel. Sheet” на “Excel. Chart”. Для вставлення об'єкту з буферу обміну в програмі Visual Basic слідує задати для властивості Action елементу керування значення acOLEPaste або acOLEPasteSpecialDlg.

Властивості **Класс OLE (OLEClass)** і **Класс (Class)** схожі, але не тотожні. В значенні властивості Клас OLE (OLEClass) міститься загальний опис об'єкту OLE, тоді як значенням властивості Клас (Class) є ім'я, що слід використовувати для посилання на цей об'єкт OLE в програмі Visual Basic. Прикладами значень властивості Клас OLE (OLEClass) можуть служити “Діаграма Microsoft Excel”, “Документ Microsoft Word” і “Точечный рисунок Paintbrush”.

Наступна процедура, властива для кнопки, створює при натиску кнопки зв'язаний об'єкт OLE і задає змінні розміри об'єкту, що дозволять повністю відобразити його зміст в елементі керування.

```
Sub Кнопка1_Click
OLE1_Class= “Excel. Sheet”                  ‘Задає ім'я класу
OLE1. OLETypeAllowed= acOLELinked           ‘Задає Тип об'єкту
OLE1. SourceDoc= “C:\Excel\Oletext. xls”    ‘Задає файл-джерело
OLE1. SourceItem= “R1C1: R5C5”              ‘Задає зв'язуємий фрагмент
OLE1. Action= acOLECreateLink               ‘Створює зв'язаний об'єкт
OLE1.SizeMode= acOLESizeZoom                ‘Задає змінні розміри елементу керування
End Sub
```

PictureData використовується для копіювання малюнку з форми, звіту або елементу керування в інший об'єкт, що підтримує властивість Малюнок (Picture).

В якості значення властивості PictureData слід задати значення властивості PictureData вхідного елементу керування: малюнку, кнопки, вимикача форми або звіту.

Значення даної властивості задається в програмі Visual Basic.

Дана властивість дозволяє вивести форму з різними фоновими малюнками в залежності від режиму роботи користувача. Наприклад, можна відкрити форму “Клієнти” з одним фоновим малюнком для режиму введення даних і з іншим для режиму перегляду.

Крім того, властивість PictureData використовується разом з подією Таймер (Timer) і властивістю Інтервал таймера (TimerInterval) для простої анімації малюнків в формі.

OLEData копіює дані з однієї вільної рамки об'єкту в іншу або з одного елементу ActiveX в інший.

Ім'я **Елементу. OLEData**

Властивість OLEData може мати наступні значення.

Значення	Опис
Ім'я Елементу	Обов'язкове. Ім'я елементу керування типу вільної рамки об'єкту.
OLEData	Обов'язкове. Дані, що містяться в вільній рамці об'єкту або в елементі ActiveX.

Властивість OLEData доступна тільки в програмі Visual Basic. Для завдання властивості OLEData елементу ActiveX можливо використання в режимі конструктора цієї ж властивості іншого елементу ActiveX.

Дана властивість дозволяє виводити в вільній рамці об'єкту дані з іншої вільної рамки об'єкту.

При привласненні в якості значення властивості OLEData одного елементу ActiveX значення цієї ж властивості іншого елементу ActiveX перший з них стає копією другого. Наприклад, в результаті наступного привласнення значення властивості елемент керування TreeView стає елементом керування **Календарь**:

Me! TreeView. OLEData=Me! МойКалендарь. OLEData

Тип **OLE (OLEType)** дозволяє визначити, чи міститься в елементі керування об'єкт OLE, і якщо так, чи є об'єкт зв'язаним або впровадженим.

Тип OLE (OLEType) може мати наступні значення.

Значення	Описи	Константа
Зв'язаний	Елемент керування Містить зв'язаний об'єкт. Вся обробка даних об’єкта виконується в додатку, в якому об'єкт був створений.	AcOLELinked
Впроваджений	Елемент керування містить впроваджений об'єкт. Обробка даних в об'єкті виконується в додатку Access.	AcOLEEmbedded
Відсутній	Елемент керування не містить об'єкт OLE.	AcOLENone

Властивість Тип OLE (OLEType) доступна тільки для читання в усіх режимах.

При створенні об'єкту OLE слід у властивості **Допустимый тип OLE (OLETypeAllowed)** визначити типи об'єктів, що можуть бути поміщені в даний елемент керування.

Допустимий тип OLE (OLETypeAllowed) визначає типи об'єктів OLE, що можуть міститися в елементі керування.

Властивість **Допустимый тип OLE** (OLETypeAllowed) може мати наступні значення.

Значення	**Опис**	**Константа**
Зв'язаний	Елемент керування може містити тільки зв'язаний об'єкт.	AcOLELinked
Впроваджений	Елемент керування може містити тільки впроваджений об'єкт.	AcOLEEmbedded
Всі (значення за змовчанням).	Елемент керування може містити тільки зв'язаний або впроваджений об'єкт.	AcOLEEither

Значення властивості **Допустимый тип OLE** (OLETypeAllowed) задається в вікні властивостей, в макросі або в програмі Visual Basic. Можна також задати значення цієї властивості, що використовується за замовчуванням, в вікні стандартних властивостей елементу керування або за допомогою засобу DefaultControl в Visual Basic.

Контрольні питання до лабораторної роботи 5:

1. **Що представляє собою центр безпеки та конфіденційності?**
2. **Якими способами можна забезпечити захист БД?**
3. **Яким чином можна шифрувати БД?**
4. **Як скасувати пароль?**
5. **Що представляє собою ACCDE файл?**

6. **Які об’єкти називають OLE?**
7. **Які властивості мають OLE об’єкти при створенні або встановленні у Microsoft Access?**
8. **Яким чином можливо захистити модулі БД?**

Лабораторна робота №6

Тема: Інформаційні системи в мережах. Зв'язок БД та електронних таблиць. Встановлення зв'язків та публікація даних у Web.

Мета: Навчитися користуватися Microsoft Excel. Навчитися використовувати зв'язок БД та електронних таблиць. Навчитись працювати з функціями, які забезпечують взаємодію з Internet.

Зміст роботи за варіантом індивідуального завдання:

1. Побудуйте електронну або друковану форму з елементами керування.
2. Заповніть електронну форму (таблицю) необхідними даними.
3. Виконайте обчислення з своїми даними, сортування, фільтр.
4. Змініть формат або вміст існуючого шаблону.
5. Побудуйте діаграми.
6. Створіть зображення комірок з прив'язкою до вхідних даних.
7. Підключіться до таблиці БД Access в Excel.
8. Додайте гіперпосилання в таблицю своєї бази даних.
9. Додайте гіперпосилання в форму своєї бази даних, щоб зв`язати зі звітом.
10. Додайте гіперпосилання в форму, щоб зв`язати фому з зовнішнім документом Microsoft Office.
11. Опублікуйте БД в Інтернет (версія 2013 і вище, інакше створіть сторінку доступа до даних).

Теоретичні відомості:

Комірка, блок комірок, рядок, діапазон рядків, стовпець, діапазон стовпців, аркуш, книга – це основні об'єкти, з якими працює користувач Excel.

У формулах, що містять посилання (адреси комірок), використовують два види адрес: відносні і абсолютні. Адреса комірки називається відносною, якщо при зміні місця розташування комірки, що містить цю адресу, вона змінюється за тими ж правилами. Абсолютна адреса утворюється з відносної за допомогою знака $.

Створення форми

Для створення особистої форми потрібно створити лист потрібного формату, що містить потрібний текст і графіку, а після цього зберегти книгу, як шаблон. Для отримання порожньої копії даної форми слід створити нову форму на основі цього шаблону.

При заповненні форми в Microsoft Excel можна автоматизувати введення і аналіз даних.

Щоб оглянути приклади форм, які можна створити в Microsoft Excel, виберіть кнопку **Создать** в меню **Файл** і створіть нову книгу з шаблону замовлення на купівлю, рахунки або розрахунки видатків. Ці вбудовані форми являють собою відформатовані листи із спеціальними елементами керування і формулами, що автоматизують заповнення форми і обробку введених даних.

Для цього:

Відформатуйте осередки аркуша.

Для додавання прапорців, перемикачів і інших елементів керування до форми, що буде використовуватися в Microsoft Excel, існують елементи керування на панелі інструментів **Формы**. (Якщо форма буде використовуватися в спеціальній програмі Visual Basic для додатків, зв'язаної з іншими програмами Office, Інтернетом або Web, елементи керування додаються за допомогою панелі інструментів Елементи керування.)
Для відвертання небажаних змін електронної форми можна дозволити користувачам змінювати і вводити дані тільки в означені комірки.
Вилучіть всі невикористані листи з книги.
Після завершення укладання форми виберіть команду **Сохранить как, Шаблон** (*. xlt).
В полі **Папка** виберіть папку, в якій повинен зберігатися даний шаблон.

Для створення і заповнення порожніх копій даної форми, основаної на шаблоні, слід вибрати команду **Создать**, а потім — потрібний шаблон форми. Для збереження кожного набору даних, введених в форму, в якості запису бази даних слід зв'язати осередки форми з полями бази даних рис. 1.

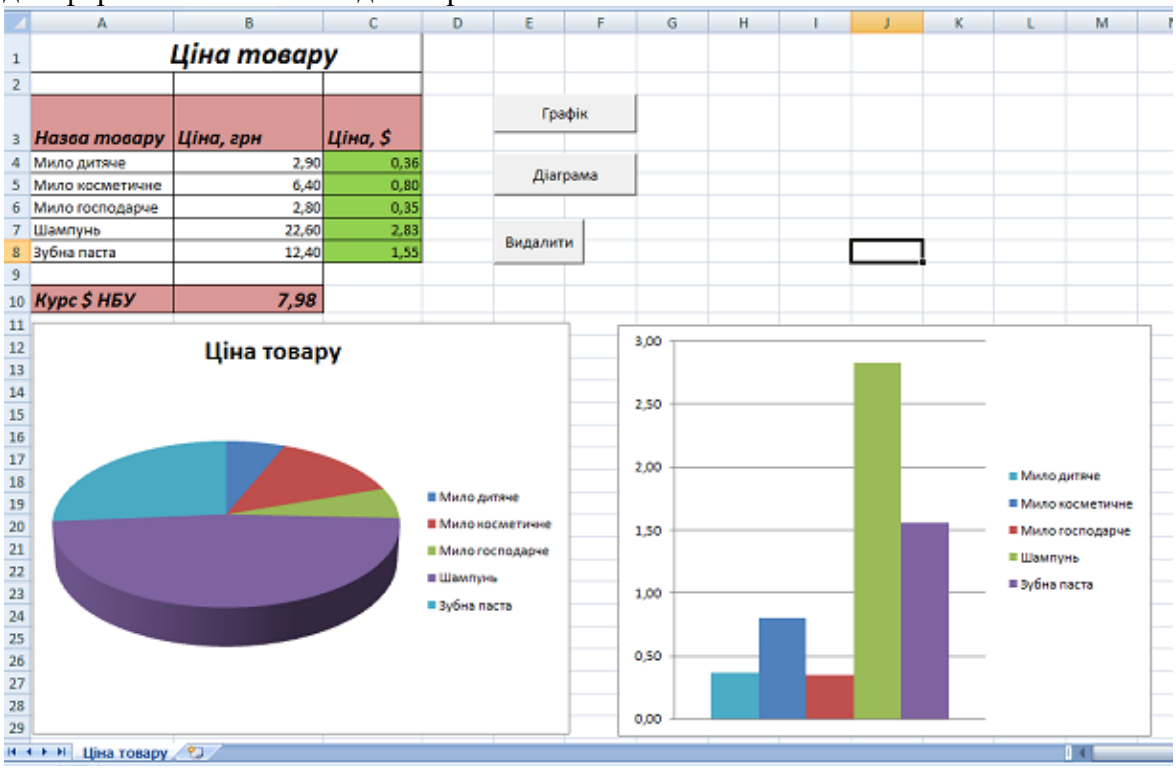

Рисунок 1 – Заповнена форма

Зміна формату або вмісту існуючого шаблону

Для виконання цього необхідно:

1 Натисніть кнопку **Открыть**, а потім відкрийте шаблон, що необхідно змінити. Шаблони звичайно зберігаються в папці «Шаблони», яка знаходиться в папці Office або Microsoft Excel, в папці XLStart або в іншій додатковій папці автозагрузки.

2 Внесіть необхідні зміни в вміст, формат, макроси і інші параметри шаблону.

3 Натисніть кнопку **Сохранить**.

Ці зміни впливають тільки на нові книги або листи, створені після зміни шаблону. На наступні книги або таблиці, основані на попередній версії шаблону, жодного впливу не виявляється.

Створення діаграми

Діаграми зв'язані з даними листа, на основі яких вони були створені, і змінюються кожний раз, коли змінюються дані на аркуші.

Діаграми можуть використовувати дані несміжних комірок. Діаграма може також використовувати дані вільної таблиці.

Можна створити або впроваджену діаграму, або аркуш діаграми. Для цього потрібно:

1 Виділити осередки, що містять дані, що повинні бути відбиті на діаграмі.

2 На вкладці Вставити в групі Діаграми виконайте одну з таких дій. Виберіть тип і підтип діаграми.

3 За замовчуванням діаграма додається на лист як впроваджена діаграма. Щоб помістити діаграму на окремий лист діаграми, змініть її розташування.

Щоб швидко створити діаграму, засновану на типі діаграми, створеному за замовчанням, виділіть потрібні дані і натисніть клавіші ALT + F1 або F11. При натисканні клавіш ALT + F1 створюється впроваджена діаграма, при натисканні клавіші F11 діаграма відображається на окремому аркуші діаграми.

Використання об'єктів (комірок, форм, діаграм) в інших додатках (MS Word, PowerPoint та ін.)

Є можливість додавання даних аркуша Excel, діаграми або інших об'єктів для використання в якості ілюстрацій на арккуші або в документі іншого додатку. Можна змінювати розміри цих зображень, а також переміщати і змінювати самі зображення, подібно будь-якому іншому намальованому об'єкту.

Щоб зображення комірок аркуша змінювалося разом з даними джерела, можна встановити зв'язок об’єкту з аркушем джерела даних. В цьому випадку, наприклад, при зміні нумерації вхідних комірок, автоматично будуть відбуватись зміни в об’єкті.

Встановити зв'язок з зображенням діаграми не можна. Для автоматичної зміни діаграми при зміні вхідних даних слід виконати копіювання діаграми і вставлення об'єкту діаграми в кінцевий файл. Зображення може копіюватися як в форматі малюнку, так і в форматі растрового малюнку.

Створення зображення комірок з прив’язкою до вхідних даних

В зображення включаються тільки видимі дані комірок. При необхідності змініть розмір стовпців Microsoft Excel, щоб всі дані, що повинні бути присутніми на зображенні, стали видимі. Якщо лінії сітки видимі, вони також потрапляють на зображення.

1 Виділіть комірки аркуша, що повинні бути на зображенні.

2 Натисніть кнопку **Копировать**.

3 Натисніть лист або документ, в який буде вставлятися зображення.

4 Для копіювання на інший лист виберіть команду вкладка Главна, из **Вставить вибрать Как рисунок, Вставить связь с рисунком**. При цьому буде встановлений зв'язок зображення шляхом формули, що посилається на копійовані комірки.

Щоб скопіювати зображення в документ, створений за допомогою іншого додатку Office, скористуйтесь командою **Специальная вставка.** Виберіть параметр зв'язати, в списку **Как - Рисунок** і натисніть кнопку **ОК**.

Для зміни зображення може використовуватися панель інструментів **Настройка изображения.**

Існує кілька способів обміну даними між Microsoft Office Access і Microsoft Office Excel.

- Перенести дані до Excel з Access можна одним із трьох способів: скопіювавши їх із таблиці Access і вставивши на аркуш Excel, підключившись до бази даних Access із аркуша Excel або експортувавши до нього дані з Access.

- Щоб перенести дані до Access з Excel, можна скопіювати їх з аркуша Excel і вставити в таблицю Access, імпортувати аркуш Excel до таблиці Access або створити посилання на аркуш Excel у таблиці Access.

Підключення до даних Access в Excel

Щоб забезпечити оновлення перенесених даних Access в Excel, можна створити підключення до бази даних Access, яке зазвичай зберігається у файлі підключення до даних Office (ODC), і одержувати всі дані з таблиці або запиту. Перевагою такого підходу над імпортуванням є можливість періодично аналізувати дані в Excel без потреби постійно копіювати або експортувати їх з Access. Після підключення до даних можна автоматично оновлювати книги Excel із вихідної бази даних Access кожного разу під час поповнення її новою інформацією.

1. Клацніть клітинку, до якої потрібно помістити дані з бази даних Access.
2. На вкладці **Дані** у групі **Зовнішні дані** виберіть команду **З Access** (рис. 2).

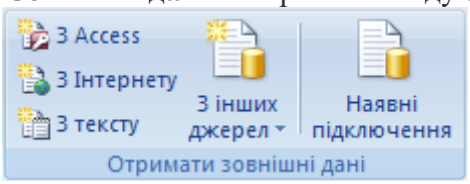

Рисунок 2

3. У списку **Шукати в** знайдіть і двічі клацніть базу даних Access, яку потрібно імпортувати. У діалоговому вікні **Вибір таблиці** виберіть потрібний об’єкт.
4. У діалоговому вікні **Імпорт даних** виконайте такі дії:

У групі **Виберіть режим перегляду цих даних та ін.**

Копіювання даних з Excel до Access

1. Запустіть Excel і відкрийте аркуш із даними, які потрібно скопіювати.
2. Виберіть рядки для копіювання.
3. На вкладці **Основне** в групі **Буфер обміну** натисніть кнопку **Копіювати**  Сполучення клавіш Можна також скористатися сполученням клавіш CTRL+C.
4. Запустіть Access і відкрийте таблицю, запит, або форму, в які потрібно вставити рядки.
5. На вкладці **Режим таблиці** у групі **Подання** натисніть команду **Вигляд** і виберіть пункт (рис. 3) **Подання таблиці**.

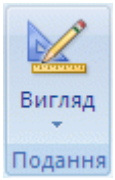

Рисунок 3

- Виконайте: щоб замінити записи, виберіть ці записи та на вкладці **основне** у групі **буфер** обміну натисніть кнопку **Вставити**  . Можна також скористатися сполученням клавіш CTRL+V. Щоб додати дані як нові записи, на вкладці **Основне** у групі **Буфер обміну** виберіть команду **Доповнити** з меню **Редагувати**.

Access та Internet

За допомогою вбудованої в Microsoft Office Web технології Access може посилати, отримувати та опубліковувати об’єкти бази даних в локальному та глобальному масштабі.

В цій роботі користуються гіперпосиланнями, щоб зв’язати базу даних з різними об’єктами, публікується об’єкт БД як Web-сторінка та створюється інтерактивна сторінка доступу.

В традиційних мережах для ідентифікації віддалених даних використовується формат UNC (Universal Naming Convention). Формат UNC визначає місце файлу на жорсткому диску комп'ютера або в локальній мережі і має вигляд: \\server\\share\path\filename.

В Web адресація даних виконується за допомогою гіперпосилань, які використовують формат URL (Uniform Resource Locator), який задає як місцезнаходження даних, так і метод доступу (протокол), та має вигляд: protocol://address//path/filename. Microsoft Office розпізнає обидва типи адресації. Гіперпосилання представляють собою окремий тип даних в Access та можуть зберігатися в таблицях, вказуватися в формах та звітах, полегшуючи взаємодію бази даних з іншими джерелами інформації, в тому числі з Інтернет.

Microsoft Access дозволяє зберігати та експортувати файли в форматі HTML. Тому при наявності зв'язку з сервером Інтернет любий об'єкт бази даних може бути перетворений у формат HTML та опублікований як Web-документ в Інтернеті.

Додавання гіперпосилань в базу даних

Простіше всього додавати гіперпосилання в базу даних, визначивши тип відповідного поля як **Гиперссылку** (Hyperlink). Будь-яке значення, яке вводиться в таке поле, буде автоматично перетворене в гіперпосилання.

Використаємо гіперпосилання, щоб зв'язати форму **Набори** з трьома джерелами інформації: звітом в базі даних **Цукерки**, зовнішнім документом Microsoft Office і адресатом електронної пошти.

Зв'язування форми зі звітом

Скористаємось гіперпосиланням, щоб напряму зв'язати форму **Набори** із звітом **Продажі по замовникам.**

1. Необхідно відкрити форму **Набори** в режимі **Конструктор** (Design).
2. На панелі інструментів конструктора форм вибираємо елемент **Вставить гиперссылку** (Insert Hyperlink). З'явиться діалогове вікно (рис. 1).

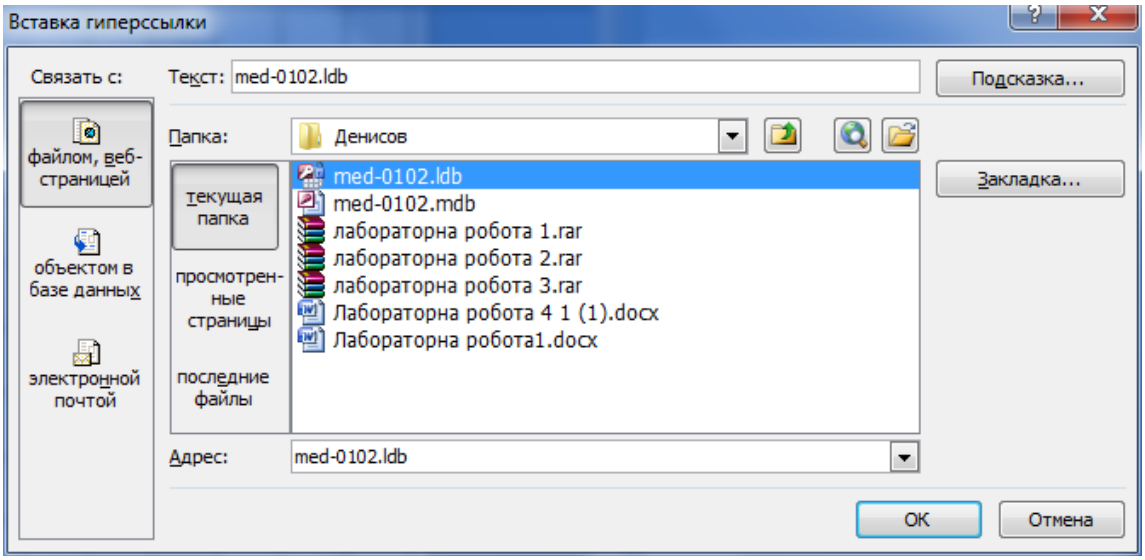

Рисунок 1

3. В діалоговому вікні на панелі **Связать с** (Link To) щелкните на пункті **Объектом в базе данных** (Object In This Database).
4. В списку об'єктів бази даних виділіть необхідний звіт (*Продажі по замовникам).*
5. **ОК**. В верхньому лівому кутку області форми **Данные** (Detail) з'явиться гіперпосилання **Продажі по замовникам**.

6. Вкажіть на гіперпосилання **Продажі по замовникам** і, коли вказівник прийме форму долоні, перенесіть його в положення, яке вам потрібно.

Аналогічно виконується: **Зв'язування форми з документом Microsoft Office, Зв'язування форми з адресатом електронної пошти** (якщо встановлений Microsoft Outlook).

Сторінка доступу до даних

Сторінки доступу мають формат HTML. Вони створюються в Access до версії 2007 і відображаються в Web-браузері. Окрім того, що сторінки доступу дозволяють відображати об'єкти бази даних в Web, вони підтримують VBScript и JavaScript, що дозволяє програмувати на одній з цих мов, знаходячись в звичному середовищі конструктора Access.

Подібно формам та звітам, сторінку доступу можна створювати в режимі конструктора, починаючи з початку. Однак простіше скористатись **Мастером страниц** або існуючою Web-сторінкою в якості основи, а потім модифікувати її, користуючись конструктором сторінок доступу.

Робота із сторінкою доступу виконується аналогічно роботі у середовищі конструктора Access, а елементи керування, розміщені на сторінці доступу (ActiveX та елементи HTML), за формою та призначенням схожі з елементами керування, які використовуються в формах Access.

Програма Access 2010 і служби Access Services – новий компонент SharePoint – можна використовувати для створення застосунків веб-баз даних. Це дає змогу:

- захищати доступ до даних і керувати ним;
- спільно використовувати дані в організації або в Інтернеті;
- створювати застосунки баз даних, для використання яких не потрібна програма Access.

Працювати з БД можуть користувачі, що мають облікові записи SharePoint, через веб-браузер. Під час публікування веб-бази даних служби Access Services створюють сайт SharePoint, на якому розміщується БД. Усі об'єкти та дані БД переміщуються до списків SharePoint на цьому сайті.

Контрольні питання до лабораторної роботи 6:

1. **Які основні об'єкти, з якими працює користувач Excel?**
2. **Яка адреса називається відносною?**
3. **Яка адреса називається абсолютною адресою?**
4. **Як автоматизувати обчислення в Excel?**
5. **Етапи створення електронної форми.**
6. **Як змінити шаблон?**
7. **Які види діаграм ви знаєте?**
8. **Як у Excel отримати дані з Access?**
9. **Які функції Access забезпечують взаємодію з Інтернет?**
10. **Які типи адресації розпізнає Microsoft Office?**
11. **Що представляють собою гіперпосилання в Access?**
12. **Яким чином за допомогою гіперпосилань зв'язати форму із звітом у своїй базі даних?**
13. **Яким чином за допомогою гіперпосилань зв'язати форму з зовнішнім джерелом?**
14. **Що розуміють під сторінкою доступу?**

Використана література

1. А.В. Маркин Построение запросов и програмирование на SQL: учеб. пособие, - Рязань: РГРТУ, 2008.-312с.
2. Ицик Бен-Ган Microsoft SQL Server 2008. Основы T-SQL:Пер. с англ.-СПб.: БХВ-Петербург, 2009.-432 с.:ил
3. Бьюли А. Изучаем SQL. – Пер. с англ.-СПб:Символ-Плюс, 2007.-312с., ил.
4. Access 2000 Шаг за шагом издательство Эком Москва, 2002.
5. С.В. Глушаков Д.В. Ломотько Базы данных Учебный курс Харьков «Фолио» Москва «АСТ» 2001.
6. І.Т. Зарецька Б.Г. Колодяжний А.М. Гуржій О.Ю Соколов Інформатика, Київ, “Навчальна книга”2002.
7. Конноллн, Томас, Бегг, Карелии. К64 Базы данных. Проектирование, реализация и сопровождение. Теория и практика. 3-е издание. : Пер. с англ. — М. : Издательский дом "Вильяме", 2003. — 1440 с. : ил.
8. А.Ю. Берко О.Н. Верес Організація баз даних Практичний курс. Навчальний посібник. Львів, 2003.
9. Боб Вилльариал Программирование Access 2002 в примерах: Пер. с англ. -
10. М.: КУДИЦ-ОБРАЗ, 2003. - 496 с.
11. Л. М. Дибкова Інформатика і комп’ютерна техніка Навчальний посібник 3-тє видання, доповнене Київ «Академвидав» 2011.
12. Гайна Г.А. Г12 Основи проектування баз даних: Навчальний посібник. − К.: КНУБА, 2005. – 204 с. ISBN 966-627-117-6
13. Microsoft [Електронний ресурс]. – Режим доступу: https://www.office.com/