МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД
«ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ»
КАФЕДРА ПРИКЛАДНОЇ МАТЕМАТИКИ ТА ІНФОРМАТИКИ

Конспект лекцій
з дисципліни «Проектування інформаційних систем»
для студентів спеціальності 122 Комп’ютерні науки
денної та заочної форми навчання

Покровськ, 2019

УДК 004.652.4

Конспект лекцій з дисципліни «Проектування інформаційних систем» для студентів спеціальності 122 Комп’ютерні науки денної та заочної форми навчання / [укл. Ярош І.В., Т.О. Черняк]. – Покровськ: ДВНЗ «ДонНТУ», 2019. – 50 с.

Конспект лекцій складений відповідно до вимог державних освітніх стандартів і рекомендований для студентів, які здобувають освітній ступень «бакалавр» освітньо-професійної програми підготовки зі спеціальності 122 Комп’ютерні науки.

Укладачі: І.В. Ярош, старший викладач кафедри ПМІ
Т.О. Черняк, асистент кафедри ПМІ

Рецензія: С.О. Ковальов, доцент, к.т.н., декан ФКНТ

Відповідальний за випуск: О.А. Дмитрієва, зав. каф. ПМІ

Затверджено навчально-методичним відділом ДВНЗ «ДонНТУ»,
протокол № 7 від 26.02.2019 р.
Розглянуто на засіданні кафедри прикладної математики та інформатики,
протокол № 8 від 24.01.2019 р.

ЗМІСТ

ВСТУП

В даний час дуже багатьом необхідні знання про бази даних і інформаційні системи. Уже давно ми, стоячи в касі з продажу квитків на потяги і літаки не дивуємося діям касирів, які недавно здавалися незвичайними, і терпляче чекаємо «вироку», який формується в великому «мозку комп'ютера» – знайде він нам квиток або не знайде. При покупці продуктів в магазині касир зчитує вартість товарів за допомогою сканера, а підсумкову суму і здачу обчислює встановлене програне забезпечення – залишається тільки перевірити, чи правильно касир вважала цю здачу. Менеджер з продажу дізнається про продажі за день, тиждень, місяць не по телефону а, просто відкривши додаток, який вже давно був «готовий» витягти ці дані з сервера і видати в самому зрозумілому менеджеру вигляді. Бухгалтер, виписуючи вам «відрядження», заносить дані про вашу поїздку в стрічкову форму, яка тут же виписує всі необхідні папери для отримання грошей в касі. У вас, напевно, знайдуться і свої приклади подібного «спілкування» з базами даних, які частіше допомагають, чим заважають в роботі.

Якщо ви менеджер з продажу або закупівель, вам обов'язково потрібно знати, що можуть програмісти, і які можливості таять в собі системи управління базами даних, як домогтися, щоб перші надали вам якомога більше з того, на що «здатні» другі. Крім того, бази даних це засіб для ефективної роботи менеджерів. Менеджер завжди може «умовити» програміста (розробника баз даних) створити для нього «канал», за яким можна за допомогою запитів «виймати» з бази даних будь-яку інформацію.

З усіх систем Windows-програмування Microsoft Visual Studio 20012 – найбільш зручний, простий і ефективний засіб для розробки інтерфейсу з базами даних. Ще більші можливості відкриваються при використанні Microsoft SQL Server 2012, щоб отримати доступ до інформації, що зберігається в базах даних.

Головною метою даного курсу є познайомити студентів з методами розробки в таких системах як Microsoft SQL Server 2012 та Microsoft Visual Studio 2012. Даний конспект лекцій призначений для студентів вже знайомих з програмуванням і розробкою баз даних інформаційних систем.

Конспект лекцій призначений для студентів денної й заочної форми навчання спеціальності 122 Комп'ютерні науки.

ЛЕКЦІЯ 1

ВИДИ ІНФОРМАЦІЙНИХ СИСТЕМ. ОСНОВНІ ПОНЯТТЯ ІНФОРМАЦІЙНИХ СИСТЕМ. ІСТОРІЯ MICROSOFT SQL SERVER 2012

1. Види інформаційних систем
2. Основні поняття інформаційних систем
3. Історія Microsoft SQL Server 2012, його версії і системні вимоги

1. Види інформаційних систем

Інформаційні системи – це комплекс засобів, призначених для зберігання, упорядкування та аналізу великих обсягів інформації.

Інформаційні системи бувають електронними і не електронними. До неелектронним інформаційних систем відносяться:

- каталог в бібліотеці;
- реєстратура в лікарні;
- бібліотека.

До електронних інформаційних систем відносяться:

- база даних відділу кадрів підприємства;
- нотатки в мобільному телефоні;
- мережа Інтернет.

Існує три види інформаційних систем:

1) **База даних** – система для зберігання великих обсягів структурованої інформації (інформації, яка вводитися за шаблоном) певного типу. До баз даних відносяться наступні інформаційні системи:

- каталог бібліотеки;
- реєстратура лікарні;
- записна книжка мобільного телефону;
- база даних відділу кадрів.

2) **База знань** – система для зберігання великого обсягу неструктурованої інформації різних типів. До баз даних відносяться наступні інформаційні системи:

- бібліотека;
- мережа Інтернет.

3) **Інформаційно-аналітична система** – система, призначена як для зберігання, так і для аналізу інформації, що зберігається (Excel, STATISTICA, SPSS).

Всі електронні інформаційні системи діляться на два класи за способом зберігання інформації:

1. Не мережеві інформаційні системи, що працюють за технологією файл-сервер. Дані системи працюють на окремому комп'ютері, без використання комп'ютерної мережі (Excel, STATISTICA, SPSS);

2. Мережеві інформаційні системи, що працюють за технологією клієнт-сервер. Дані системи працюють на комп'ютері, підключеному до комп'ютерної мережі (Інтернет).

Основна відмінність технології клієнт-сервер від технології файл-сервер полягає в способі зберігання інформації, суть технології файл-сервер полягає в наступному – інтерфейс інформаційної системи і дані, з якими вона працює зберігаються на одному комп'ютері (локально).

Зауваження:

1) Клієнтами мережі є комп'ютери користувачів, підключені до мережі. Клієнти отримують доступ до сервера через мережу. Іноді клієнти мережі називають клієнтськими комп'ютерами.

2) **Сервер мережі** – комп'ютер, який управляє мережею. Всі ресурси сервера доступні клієнтам мережі, тобто будь-які зміни даних на сервері відразу видно всім клієнтам мережі.

В інформаційних системах, побудованих за технологією клієнт-сервер, інформація зберігається на сервері, а інтерфейс інформаційної системи зберігається на клієнтських комп'ютерах, через нього користувачі інформаційної системи отримують доступ до даних.

<u>Переваги та недоліки технології Файл-Сервер:</u>

+ простота розробки;

+ незалежність від комп'ютерної мережі;

+ високий захист від несанкціонованого доступу;

- не оперативне оновлення даних на декількох комп'ютерах;

- висока вартість комп'ютерів для роботи в такій системі;

- складність зміни структури даних.

<u>Переваги та недоліки технології Клієнт-Сервер:</u>

+ проста синхронізація даних;

+ низька вартість апаратного забезпечення (потужним повинен бути тільки сервер);

+ оперативна зміна структури даних;

- низький захист від несанкціонованого доступу;

- залежність від комп'ютерної мережі;

- висока вартість.

2. Основні поняття інформаційних систем

Будь-яка інформаційна система або база даних (з точки зору їх створення) в мовах програмування складаються з трьох компонентів:

1. **Файл даних** – файл, що знаходиться на локальному комп'ютері або на сервері, який містить в собі структуру даних. До структури даних відносяться таблиці, запити і фільтри, а також процедури, що зберігаються, які призначені для користувача функції, діаграми і тригери.

2. **Об'єкт зв'язку** – об'єкт мови програмування, який здійснює зв'язок між файлом даних і інтерфейсом інформаційної системи.

3. **Інтерфейс інформаційної системи** – комплекс засобів, який здійснює взаємодію системи з кінцевими користувачами. Він може знаходитися як на клієнтському комп'ютері, так і на сервері.

Розробка ІС за технологією клієнт-сервер складається з декількох етапів:

1. На сервер в комп'ютерній мережі встановлюється серверна СУБД (наприклад, Microsoft SQL Server, My SQL, Oracle), встановлюється серверна частина СУБД. Якщо реалізується web-інтерфейс, то на сервер ставитися програма web-сервер (наприклад, Apache).

2. Якщо реалізуються клієнтські програми, то на всі клієнтські частини мережі ставитися клієнтська частина (даний крок не обов'язковий і виконується тільки в тому випадку, якщо користувачі інформаційної системи мають можливість управляти сервером).

3. Налаштовується серверна частина СУБД, клієнтські частини СУБД і web-сервер.

4. Визначається структура даних (зв'язки між таблицями і типи даних полів), також визначаються первинні і вторинні таблиці в запитах.

5. На сервері створюються таблиці і запити, що виконуються на стороні сервера. Перед створенням запитів, таблиці заповнюються початковими даними. Також створюються збережені процедури, призначені для користувача функції, діаграми і тригери.

6. У разі використання клієнтського застосування, за допомогою мови програмування створюються об'єкти зв'язку, вони підключаються до таблиць, запитів і збережених процедур. Також на них створюються запити і процедури, що виконуються на стороні серверу.

7. Створюються форми.

8. Створюються звіти.

9. Система заповнюється реальними даними.

Зауваження: при створенні і заповненні таблиць інформаційної системи необхідно слідувати 3 правилам:

1) У таблицях не повинно бути повторюваних груп записів. Це досягається введенням індексних полів, тобто сортуванням записів.

2) У таблиці не повинно бути полів з однаковими іменами. Це досягається розбивкою однієї таблиці на декілька, з подальшим зв'язуванням їх запитом.

3) Не повинно бути правил при заповненні таблиць, це досягається хаотичністю заповнення таблиць бази даних.

Інформаційна система, яка задовольняє цим умовам, називається **нормалізованою інформаційною системою** або базою даних.

3. Історія Microsoft SQL Server 2012, його версії і системні вимоги

Родоначальником серії SQL Server і його основою є мова запитів SQL. Дана мова була створена компанією IBM на початку 1970р. минулого століття. Спочатку він називався SEQVEL (Structured English Query Language) В основу мови SQL, що використовується в SQL Server, лягла різновидність мови T-SQL (Transact-SQL).

На початку 80г. фірма IBM і зокрема в той час її підрозділами Microsoft і Sybase створюється перша версія мережевої СУБД, яка називалася SQL Server версія 1.0, для операційної системи IBM OS/2. Після цього під цю операційну систему було випущено ще 3 версії SQL Server.

В середині 80-х років компанія Microsoft і Sybase відокремлюються від фірми IBM, і Microsoft починає роботу над своєю операційною системою Windows, і разом з компанією Sybase починає розвиток SQL Server.

В середині 90-х років (зокрема в 1995р.) Microsoft створила операційну систему Windows NT і разом з компанією Sybase випускає першу версію SQL Server для Windows версії 4.1.

Після цього компанія Sybase розриває свої відносини з Microsoft і Microsoft створює Microsoft SQL Server 6.0. Дана версія була призначена для роботи в операційній системі Windows NT 95і 98.

У 1999р. виходить версія Microsoft SQL Server 7.0, яка стала однією з найпопулярніших серверних СУБД в світі. У 2000 р. виходить 8-я версія Microsoft SQL Server 2000. У 2005 році виходить нова версія сервера, заснована на новій технології NET, а в 2008 році виходить її поліпшена версія Microsoft SQL Server 2008.

Література [1, 2]

ЛЕКЦІЯ 2

ОСНОВНІ КОМПОНЕНТИ MICROSOFT SQL SERVER 2012. СТВОРЕННЯ ФАЙЛУ ДАНИХ. УПРАВЛІННЯ БАЗАМИ ДАНИХ ЗА ДОПОМОГОЮ КОМАНД МОВИ T-SQL

1. Основні компоненти Microsoft SQL Server 2012
2. Створення файлу даних
3. Управління базами даних за допомогою команд мови T-SQL

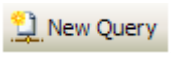
1. Основні компоненти Microsoft SQL Server 2012

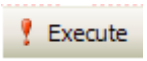
Всі компоненти Microsoft SQL Server 2012 запускаються з меню «Пуск \ Програми \ Microsoft SQL Server 2012. У Microsoft SQL Server 2012 складається з наступних компонентів:

1. Deployment Wizard – майстер з виведення інформації, що зберігається на сервері.
2. SQL Server Installation Center – центр установки SQL Server 2012.
3. Reporting Services Configuration Manager – менеджер служби настройки звітів.
4. SQL Server Configuration Manager – менеджер настройки серверу.
5. SQL Server Error and Usage Reporting – служба протоколювання роботи сервера і служба звітів про помилки.
6. Microsoft Samples Overview – посилання на сайт корпорації Microsoft, де можна переглянути приклади роботи з сервером.
7. SQL Server Books Online – повна довідкова система по Microsoft SQL Server 2012. Вона містить довідки, як з програмування, так і по адмініструванню сервера.
8. SQL Server Tutorials – підручники по роботі з сервером.
9. Data Profile Viewer – перегляд профілів по роботі з даними.
10. Execute Package Utility – інструменти зі стиснення даних.
11. Database Engine Tuning Advisor – майстер настройки ядра бази даних.
12. SQL Server Profiler – настройка профілів по роботі з даними.
13. Import and Export Data – імпорт і експорт даних;
14. SQL Server Business Intelligence Development Studio – інтегроване середовище розробки Business Intelligence Development Studio;
15. SQL Server Management Studio – графічна оболонка для управління сервером і розробки баз даних.

2. Створення файлу даних

Нову БД можна створити, використовуючи стандартні команди мови T-SQL. Для створення нової БД необхідно зробити активну БД «Master». Це можна зробити або вибором її зі списку БД на панелі інструментів, або набором команди USE Master на вкладці нового запиту.

Зауваження: усі команди мови T-SQL набираються на вкладці нового запиту (SQLQuery). Для того щоб створити новий запит на панелі інструментів необхідно натиснути кнопку 

. Для виконання команд мови T-SQL на панелі інструментів необхідно натиснути кнопку 

або на вкладці нового запиту набрати команду GO.

У Microsoft SQL Server БД складається з двох частин:

- **файл даних** – файл, що має розширення mdf і де знаходяться всі таблиці і запити;
- **файл журналу транзакцій** – файл, що має розширення ldf, містить журнал, де фіксуються всі дії з БД. Даний файл призначений для відновлення БД у разі її виходу з ладу.

Для створення нового файлу даних використовується команда CREATE DATABASE, яка має наступний синтаксис:

```
CREATE DATABASE <Ім'я БД>
(Name = <Логічне ім'я>, FileName = <Файл> [Size = <Поч.розмір>,]
[Maxsixe = <Макс.розмір>,] [FileGrowth = <Крок>])
[LOG ON
(Name = <Логічне ім'я>, FileName = <Файл> [Size = <Поч.розмір>,]
[Maxsixe = <Макс.розмір>,] [FileGrowth = <Крок>])
```

Тут Ім'я БД – ім'я створюваної БД;

Логічне ім'я – визначає логічне ім'я файлу даних БД, за яким відбувається звернення до файлу даних.

Файл – визначає повний шлях до файлу даних.

Поч.розмір – початковий розмір файлу даних в Мб.

Макс.розмір – максимальний розмір файлу даних в Мб.

Крок – крок збільшення файлу даних, або в Мб або в%.

Параметри в розділі `LOG ON` аналогічні параметрам в розділі `CREATE DATABASE`. Однак вони визначають параметри журналу транзакцій.

Приклад: Створити БД «Students», розташовану в файлі «D:\Students.mdf» і має початковий розмір файлу даних 1мб., Максимальний розмір файлу даних 100мб. і крок збільшення файлу даних рівний 1мб. Файл журналу транзакцій

даної БД має ім’я «StudentsLog» і розташований в файлі «D:\Students.ldf». Даний файл має початковий розмір рівний 1мб., Максимальний розмір рівний 100мб. і крок збільшення рівний 1мб.

```
CREATE DATABASE Students (Name = Students,
FileName = ‘D: \ Students.mdf’, Size = 1Mb,
Maxsixe = 100Mb, FileGrowth = 1Mb) LOG ON
(Name = StudentsLog,
FileName = ‘D: \ Students.ldf’, Size = 1Mb,
Maxsixe = 20Mb, FileGrowth = 1Mb)
```

3. Управління базами даних за допомогою команд мови T-SQL

У мові запитів T-SQL з БД можливі наступні дії:

1. Відображення відомостей про БД: `EXEC sp_helpdb <Ім’я БД>`.

2. Зміна параметрів БД: `EXEC sp_dboption <Ім’я БД>,<Параметр>, <Значення>`.

3. Додавання нових файлів, видалення файлів і перейменування файлів, що входять в БД:

```
ALTER DATABASE <Ім’я БД>
ADD FILE (<Параметри>) |
REMOVE FILE <Логічне ім’я файлу> | MODIFY FILE (<Параметри>)
```

де, розділ `ADD FILE` – додає файл, `REMOVE FILE` – видаляє, а розділ `MODIFY FILE` – змінює параметри файлу.

4. Стиснення всієї БД: `DBCC SHRINKDATABASE <Ім’я БД>`.

5. Стиснення конкретного файлу БД: `DBCC SHRINKFILE <Логічне ім’я файлу>`.

6. Перейменування БД: `EXEC CP_RENAMEDB <Ім’я БД>,<Нове ім’я БД>`.

7. Видалення БД: `DROP DATABASE <Ім’я БД>`.

Зауваження: вищеперелiковані команди використовують такі параметри:

- `<Ім’я БД>` – ім’я БД з якою відбувається дія;
- `<Параметр>` – змінний параметр;
- `<Значення>` – нове значення змінюваного параметра;
- `<Параметри>` – параметри файлу БД, аналогічні параметрам, використовуються в команді `CREATE DATABASE`;
- `<Логічне ім’я файлу>` – логічне ім’я файлу, що входить в БД;
- `<Нове ім’я БД>` – нове ім’я БД.

Література: [1, 3].

ЛЕКЦІЯ 3

ТАБЛИЦІ. ТИПИ ДАНИХ І ВЛАСТИВОСТІ ПОЛІВ. СТВОРЕННЯ І ЗАПОВНЕННЯ ТАБЛИЦЬ

1. Таблиці. Типи даних полів
2. Створення таблиць
3. Заповнення таблиць
4. Видалення окремих стовпців і окремих рядків з таблиці
5. Зміна даних в таблиці

1. Таблиці. Типи даних полів

Вся інформація в базі даних зберігається в таблицях. Таблиці це звичайні таблиці для зберігання даних. Таблиці складаються із записів.

Запис це рядок в таблиці. Вся інформація обробляється по записах.

Кожен запис складається з полів. Поле це стовпець таблиці. Кожне поле має три характеристики:

- ім’я поля – використовується для звернення до поля;
- значення поля – визначає інформацію, збережену в полі;
- тип даних поля – визначає який вид інформації можна зберігати в полі.

У SQL сервер використовується наступні типи даних:

1. **Бітові типи даних** які містять послідовності нулів і одиниць: Binary (n) і Varbinary(n), де n довжина. Вміст полів типу Binary завжди одно n, різниця заповнюється пробілами. Varbinary розмір поля дорівнює n або більшого.

2. **Цілочисельні типи даних** – типи даних для зберігання цілих чисел (в дужках вказано діапазон значень типу даних): Tinyint(0-255), Smallint(± 32000), Int(± 2000000000), Bigint(± 2 63).

3. **Типи даних для зберігання дрібних чисел:** Real сім знаків після коми, Float(m) може зберігати числа з m знаків, максимальне m = 38, Decimal(mn) дробові числа з m знаків до коми і n після.

4. **Спеціальні типи даних:** Bit – логічний тип даних, являється заміною логічного типу Boolean в Visual Basic, Text – тип для зберігання великих обсягів тексту, одне поле може зберігати до 2 Гб тексту, Image – тип даних для зберігання до 2 Гб малюнків, RowGUID – унікальний ідентифікатор рядка таблиці, SQL_Variant – аналогічний типу Variant в Visual Basic.

5. Типи даних дати і часу: Datetime (від 1.01.1953 до 3.12. 1999). SmallDatetime (від 1.01.19 до 6.07 2079).

6. Грошові типи даних для зберігання фінансової інформації: Money(± 10^{15} і 4 знаки після нуля), Smallmoney(± 20000,0000).

7. Автоматично оновлюванні типи даних - аналоги лічильників, але в цій ролі вони не використовуються: RowVersion унікальний ідентифікатор рядка. TimeStamp – закодована дата і час створення строки.

2. Створення таблиць

Для створення таблиць в SQL Server в першу чергу необхідно зробити активною ту БД, в якій створюється таблиця. Для цього можна в новому запиті набрати команду: `USE <Ім'я БД>`, або на панелі інструментів необхідно вибрати в списку, що випадає робочу БД. Після вибору БД можна створювати таблиці.

Таблиці створюються командою: `CREATE TABLE <Ім'я таблиці> (<Ім'я поля1> <Тип1> [IDENTITY NULL | NOTNULL], <Ім'я поля2> <Тип2>, ...)`

Тут `<Ім'я таблиці>` – ім'я створюваної таблиці;

`<Ім'я поля>` – імена полів таблиці;

`<Тип>` – типи полів;

`<IDENTITY NULL | NOT NULL>` – поле лічильник.

Зауваження: Якщо ім'я поля містить пробіл, то воно полягає в квадратні дужки.

Приклад: Створити таблицю «Студенти», що містить поля: Код студента (первинне поле зв'язку, лічильник), ПІБ, Адреса, Код спеціальності (вторинне поле зв'язку):

```
CREATE TABLE Студенти
([Код студенту] Bigint Identity,
ПІБ Varchar (20),
Адреса Varchar (100),
[Код спеціальності] Bigint)
```

Зауваження: Якщо необхідно створити обчислювальне поле, то в команді Create Table у вичісляємому полі замість типу даних потрібно вказати вираз.

Приклад: розрахувати середній бал студента за трьома його оцінками.

```
CREATE TABLE Оцінки
(ПІБ Varchar (20), Оценка1 int, Оценка2 int, Оценка3 int,
[Середній бал] = (Оценка1 + Оценка2 + Оценка3) / 3)
```

Зауваження: Отримання інформації про таблиці здійснюється застосуванням команди: `EXEC SP_HELP <Ім'я таблиці>`. Видалення таблиці здійснюється командою: `DROP TABLE <Ім'я таблиці>`.

3. Заповнення таблиць

У SQL Server 2012 заповнення таблиць проводиться за допомогою наступної команди:

```
INSERT <Ім'я таблиці> [(<Список полів>)] VALUES (<Значення полів>)
```

де `<Ім'я таблиці>` – таблиця, куди вводимо дані, `(<Список полів>)` – список полів, куди вводимо дані, якщо не вказуємо, то мається на увазі заповнення всіх полів, в списку полів поля вказуються через кому, `(<Значення полів>)` – значення полів через кому.

В якості значення можна вказати константу Default, тобто буде поставлено значення за умовчанням, або можна підставити оператор Select. Тут він використовується як інструмент обчислення формул.

Приклад: Додавання запису має наступні значення полів ПІБ = Іванов, Адреса = Покровськ, Код спеціальності = 122 в таблицю «Студенти».

```
INSERT Студенти (ПІБ, Адреса, [Код спеціальності])
VALUES ( 'Іванов А.А.', 'Покровськ', 122)
```

4. Видалення окремих стовпців і окремих рядків з таблиці

З таблиці можна видалити всі стовпці, або окремі записи. Це здійснюється командою

```
DELETE <Ім'я таблиці> [WHERE <Умова>]
```

де `<Умова>` – умова, яким задовольняються видаляємі записи, якщо умова не вказана, то видаляються всі стовпці таблиці. Якщо умова зазначена то видаляються записи поля яким відповідають умови.

Приклад: Видалити записи з таблиці «Студенти», у яких поле Адреса = Покровськ.

```
DELETE Студенти
WHERE Адреса = 'Покровськ'
```

5. Зміна даних в таблиці

Для цього використовується наступна команда:

```
UPDATE <Ім'я таблиці> SET
<Ім'я поля1> = <Вираз1>, [<Ім'я поля2> = <Вираз2>,]
...
[WHERE <Умова>]
```

де <Ім'я поля1 »,« Ім'я поля2> – імена змінних полів, <Вираз1>, <Вираз 2> – або конкретні значення, або NULL, або оператори SELECT. Тут SELECT застосовується як функція. <Умова> – умова, яким повинні відповідати записи, поля яких змінюємо.

Приклад: У таблиці «Студенти» у студента Іванова А.А. поміняти адресу Покровськ на Київ, а код спеціальності замість 122 поставити 121.

```
UPDATE Студенти
SET
Адреса = 'Київ', [Код спеціальності] = 121 WHERE ПІБ = 'Іванов А.А.'
```

Зауваження: в якості вираження можна використовувати математичні формули. Наприклад: `SET [Середній бал] = (Оценка1 + Оценка2 + Оценка3) / 3)` обчислює поле «Середній бал» як Середнє полів «Оценка1», «Оценка2» и «Оценка3».

При цьому поля «Оценка1», «Оценка2» і «Оценка3» повинні вже існувати і тип даних поля «Середній бал» повинен бути з плаваючою комою (Наприклад Real).

Зауваження: Якщо необхідно з таблиці видалити всі записи, але зберегти її структуру, для цього використовують команду `TRUNCATE TABLE <Ім'я таблиці>` при цьому всі дані будуть видалені, але сама таблиця залишиться.

Література: [4].

ЛЕКЦІЯ 4

СТВОРЕННЯ ЗАПИТІВ И ФІЛЬТРІВ. ОБЧИСЛЕННЯ ЗА ДОПОМОГОЮ ОПЕРАТОРА SELECT. ВБУДОВАНІ ФУНКЦІЇ

1. Створення запитів і фільтрів

2. Виконання обчислень за допомогою оператора SELECT. вбудовані функції

2.1 Математичні функції

2.2 Строкові функції

2.3 Функції дат

2.4 Системні функції

2.5 Агрегатні функції

1. Створення запитів і фільтрів

Запити призначені для зв’язку однієї або декількох таблиць, також вони можуть здійснювати відбір окремих полів з таблиці і виробляти фільтрацію даних згідно з умовою, накладену на одне або декілька полів, такі запити називають фільтрами.

Для реалізації запитів використовують спеціальну мову запитів SQL (Standard Query Language).

В ІС запити можуть перебувати як на стороні клієнтського додатку, так і на стороні серверу. Якщо запит зберігається на стороні клієнта, то він прописується всередині об’єкту зв’язку. В цьому випадку клієнтський додаток не залежить від файлу даних. Файл даних містить тільки таблиці, тому, ми легко можемо модифікувати клієнтську програму, не зачіпаючи файл даних, але в цьому випадку запит передається серверу через мережу, що може викликати проблеми з безпекою.

Якщо запит зберігається або виконується на сервері, то сам запит виступає в якості компонента БД, вся передача інформації відбувається всередині файлу даних, тобто всередині самого серверу, клієнтському додатку тільки передаються результати виконання запиту. У цьому випадку забезпечується високий захист даних, але в разі зміни запиту доведеться міняти сам файл даних.

Всі запити діляться на:

– статичні;

– динамічні.

Структура статичних запитів незмінна в ході роботи з програмою, а динамічні запити можуть змінюватися в залежності від ситуації.

Зауваження: зазвичай динамічні запити можуть бути реалізовані тільки за допомогою запитів, що виконуються на стороні клієнта. Якщо необхідно реалізувати динамічні запити, які виконуються на стороні серверу, то в цьому випадку необхідно використовувати збережені процедури.

Збережені процедури – SQL запит, що зберігається на стороні серверу і цей запит має параметри, які підставляються всередину SQL коду. При виклику процедури, що необхідно передавати всередину її значення параметрів.

В основному запит або збережена процедура або реалізує зв'язок між таблицями, або здійснює фільтрацію даних, деякі SQL запити також можуть робити обчислення.

У разі зв'язків між таблицями одна таблиця завжди виступає первинною, а інша вторинною, зв'язок відбувається за допомогою полів зв'язку. При зв'язку порівнюються записи з однаковими значеннями полів зв'язку. Первинна таблиця завжди заповнюється першою, а її поле зв'язку заповнюється автоматично (тип даних – лічильник). Вторинна таблиця завжди заповнюється після заповнення первинної таблиці, значення її поля зв'язку підставляється зі значень поля зв'язку первинної таблиці. Поля зв'язку повинні мати однаковий тип даних.

Існує чотири види зв'язку між таблицями:

1. Одна до одної - одному полю в первинній таблиці відповідає одне поле у вторинній таблиці;

2. Одна до багатьох – одному полю в первинній таблиці відповідає декілька полів у вторинній таблиці;

3. Багато до одного – декільком полям в первинній таблиці відповідає одне поле у вторинній таблиці;

4. Багато до багатьох – одному полю в первинній таблиці відповідає декілька полів у вторинній таблиці і навпаки.

Запити з першим видом зв'язку називаються простими, а з іншими видами зв'язку – складними. Якщо в БД є хоча б дві пов'язані таблиці, то БД – реляційна.

Щоб створити запит необхідно зробити активною БД для якої створюється запит, потім в робочій області редактора запитів створити запит за допомогою команди SELECT, що має наступний синтаксис:

```
SELECT [ALL|DISTINCT]
[TOP|PERCENT n]
<Список полів>
[INTO <Ім'я нової таблиці>]
[FROM <Ім'я таблиці>]
[WHERE <Умова>]
[GROUP BY <Поле>]
[ORDER BY <Поле> [ASC|DESC]]
[COMPUTE AVG|COUNT|MAX|MIN|SUM (<Вираз>)]
```

де параметри ALL|DISTINCT показують, які записи обробляються: ALL обробляє всі записи, DISTINCT тільки унікальні, видаляються повторення записів. TOP n визначає скільки записів обробляють, якщо вказано PERCENT, то n вказує відсоток від загального числа записів. <Список полів> – тут вказуються поля які відображаються з таблиць через кому.

Зауваження:

1) Якщо імена видимих полів в різних таблицях не повторюються, то ми можемо вказувати тільки імена стовпців або полів без вказівки самих полів (ПІБ, посада). Якщо відображаються поля з різних таблиць з однаковими іменами потрібно вказувати і ім'я таблиці <Ім'я поля>. <Ім'я таблиці>;

2) Тут же можна привласнювати псевдоніми полям, таким чином <Ім'я поля> AS <Ім'я користувача>

3) Якщо необхідно вивести всі поля з таблиці, то їх можна замінити позначкою «*»

Розділ INTO. Якщо присутній цей розділ, то на основі результатів запиту створюється нова таблиця. Параметр INTO це ім'я нової таблиці.

Розділ FROM. Тут вказуються таблиці і запити, через кому, які беруть участь в новому запиті.

Зауваження: У розділі FROM так само можна задавати складні зв'язки, зв'язок поля однієї таблиці, з декількома полями іншої таблиці. В цьому випадку розділ FROM матиме такий вигляд: FROM <Таблица1> INNER JOIN <Таблица2> ON <Таблица1>. <Поле1> оператор <Таблица2>. <поле2> ...
Тут встановлюється взаємозв'язок Таблиці 1 і Табліци2 по Полю1 і Полю2 в залежності від оператора порівняння. Таких розділів INNER JOIN може бути скільки завгодно.

Розділ WHERE. Даний розділ використовують для створення простих запитів, в цьому випадку в якості умови вказуємо зв'язувальні поля, або цей розділ використовують для створення фільтрів, тут вказують умови відбору. В умовах відбору ми можемо використовувати стандартні логічні оператори NOT, OR, AND.

Зауваження: У своєму стандартному вигляді запити можуть реалізовувати тільки статичні фільтри, але не динамічні. Для реалізації динамічних фільтрів використовуються збережені процедури.

Розділ GROUP BY – визначає поле для групування записів в запиті.

Розділ ORDER BY – визначає поле для сортування записів в запиті. Якщо вказано параметр ASC, то буде проводитися сортування за зростанням, якщо DESC – по спадаючій. За замовчуванням використовується сортування за зростанням.

Розділ COMPUTE дозволяє в кінці результатів виконання запиту вивести деякі підсумкові обчислення за запитом. Можливі такі види обчислень: AVG – середня параметра; COUNT – кількість значень варіанту не рівних NULL; MAX і MIN – максимальні і мінімальні значення параметру; SUM – сума всіх значень параметра, де <Вираз> – сам параметр. В якості параметрів зазвичай виступають поля таблиць, що беруть участь в запиті.

Приклад: Даний запит пов'язує дві таблиці Співробітники та Посади по полях Код. При своєму виконанні він відображає перші 20 відсотків співробітників з обох таблиць. З таблиці Співробітники відображаються всі поля, а з таблиці Посади тільки поле посаду. В кінці результатів виводиться кількість відображених співробітників.

```
SELECT TOP 20 PERCENT * .Співробітники, Посади.Посади
FROM Співробітники, Посади
WHERE Код.Співробітник = Код.Посади
COMPUTE COUNT (ФІО.Співробітники)
```

Приклад: Даний запит з таблиці Операції виводить всі записи, значення поля Місяць у яких дорівнює «Травень». Дані в результаті групуються по полю Операція і упорядковано за сумою операції. В кінці результатів запиту відображається загальна сума відібраних операцій за травень. Результати даного запиту зберігаються в таблиці «Угоди за травень».

```
SELECT ALL Операція, сума
INTO [Угоди за Травень] FROM Операції
WHERE Місяць = 'травень' GROP BY Операція ORDER BY Сума COMPUTE
SUM (Сума)
```

Зауваження: В даному запиті при позначенні назви полів таблиці явно не вказуються, тому що використовувалася тільки одна таблиця.

2. Виконання обчислень за допомогою оператора SELECT. вбудовані функції

Крім зв'язування таблиць та відбору даних оператор SELECT може використовуватися для обчислень. У цьому випадку він має синтаксис:

```
SELECT <Вираз>
```

де <Вираз> – якийсь математичний вираз чи функція. Вираз має стандартний вид (як в Visual Basic), він може включати в себе вбудовані функції сервера.

Зауваження: Ми можемо використовувати вбудовані функції і вирази в обчислюваних полях при створенні таблиць.

2.1 Математичні функції

Зауваження: Як параметри функції будемо вказувати відповідний їм тип даних.

ABC (numeric) – модуль числа;

ACOS/ASIN/ATAN (Float) – арккосинус, арксинус, арктангенс в радіанах;

COS/SIN/TAN/COT (Float) – косинус, синус, тангенс, котангенс;

CEILING (Numeric) – найменше ціле, більше або рівне параметру в дужках;

DEGREES (Numeric) – перетворює радіани в градуси;

EXP (Float) – експонента, ex;

FLOOR (Numeric) – найменше ціле менше або рівне висловом numeric;

LOG (Float) – натуральний логарифм ln;

LOG10 (Float) – десятковий логарифм log10;

PI () – число пі;

POWER (Numeric, y) – зводить вираз Numeric в ступінь y;

RADIANS (Numeric) – перетворює градуси в радіани;

RAND () – генерує випадкове число типу даних Float, розташоване між нулем і одиницею;

ROUND (Numeric, Довжина) – округлює вираз Numeric до заданої Довжини (Кількість знаків після коми);

SIGN (Numeric) – виводить знак числа +/- або нуль;

SQUARE (Float) – обчислює квадрат числа Float;

SQRT (Float) – обчислює квадратний корінь числа Float.

Приклади використання математичних функцій:

`SELECT ABC (-10)` результат 10

`SELECT SQRT (16)` результат 4

`SELECT ROUND (125.85,0)` результат 125

`SELECT POWER (2,4)` результат 16

2.2 Строкові функції

Строкові функції дозволяють робити операції з одним або декількома рядками.

‘Рядок1’ + ‘Рядок2’ приєднує Строку1 до Строки2;

ASCII (Char) – повертає ASCII код з самого лівого символу вираження Char;

CHAR (Int) – виводить символ відповідний ASCII коду в вираженні Int;

CHARINDEX (Зразок, Вираз) – виводить позицію Зразка вираження, тобто де знаходиться Зразок в вираженні;

DIFFERENCE (Вираз1, Вираженіе2) – порівнює два вирази, виводить числа від 0 до 4: 0 – вираження абсолютно різні; 4 – вираження абсолютно ідентичні. Обидва вирази типу даних Char;

LEFT (Char, Int) – виводить з рядка Char Int символи зліва;

RIGHT (Char, Int) – виводить з рядка Char Int символи праворуч;

LTRIM (Char) – видаляє з рядка Char прогалини зліва;

RTRIM (Char) – видаляє з рядка Char прогалини справа;

WCHAR (Int) – виводить вираз Int в форматі Unicode;

REPLACE (рядок1, рядок2, рядок3) – змінює в рядку1 всі елементи рядку2 на елементи рядку3;

REPLICATE (Char, Int) – повторює рядок Char Int раз;

REVERSE (Char) – виробляє інверсію рядку Char, тобто має у своєму розпорядженні символи в зворотному порядку;

SPACE (Int) – виводить Int прогалин;

STR (Float) – переводить число Float в рядок;

STUFF (Вираз1, Початок, Довжина, Вираз2) – видаляє з Виразу1 починаючи з позиції символу Початок кількість символів рівне параметру Довжина, замість них підставляє Вираз2;

SUBSTRING (Вираз, Початок, Довжина) – з Виразу виводиться рядок заданої Довжини починаючи з позиції Початок;

UNICODE (Char) – виводить код у форматі Unicode першого символу в рядку Char;

LOWERC (Char) – переводить рядок Char в маленькі букви;

UPPER (Char) – переводить рядок Char в заголовні букви. Приклади застосування строкових функцій:

`SELECT ASCII (‘G’)` результат 71.

`SELECT LOWER (‘ABC’)` результат abc.

`SELECT Right (‘ABCDE’)` результат

`CDE SELECT REVERCE (‘МИР’)` результат РИМ.

Зауваження: у всіх строкових функціях значення висловлювання на кшталт Char полягають в одинарні лапки.

2.3 Функції дат

Зауваження: в деяких функціях дат використовується так звана частина дат, яка кодується спеціальними символами:

dd – число дат (від 1 до 31);
dy – день року (число від 1 до 366);
hh – значення години (0-23);
ms – значення секунд (від 0 до 999);
mi – значення хвилин (0-59);
qq – значення (1-4).
mm – значення місяців (1-12);
ss – значення секунд (0-59);
wk – значення номерів тижнів у році;
dw – значення днів тижня, тиждень починається з неділі (1-7);
yy – значення років (1753 -999).

Функції дат призначені для роботи з датами або часом. Існують кілька наступні функції дат:

DATEADD (частина, число, date) – додає до дати date частину дати помножене на число;

DATEDIFF (частина, date1, date2) – виводить кількість частин дати між date1 і date2;

DATENAME (частина, date) – виводить символьне значення частин дати до заданої дати (назва днів тижня);

DATEPART (частина, date) – виводить числове значення частини дати з заданої дати (номер місяця);

DAY (date) – виводить кількість днів в заданій даті;

MONTH (date) – виводить кількість місяців в заданій даті;

YEAR (date) – виводить кількість років в заданій даті;

GETDATE ()– виводить поточну дату встановлену на комп'ютері.

Зауваження: Дати виводяться в Американському форматі: місяць/день/рік.

Приклади функції робіт з датами:

`SELECT DATEADD (dd, 5,11/20/07)` результат Nov/25/2007.

`SELECT DATEDIFF (dd, 11/20/07, 11/25/07)` результат 5 днів.

`SELECT DATENAME (mm, 11/20/07)` результат November.

`SELECT DATEPART (mm, 11/20/07)` результат 11.

Зауваження: у виразах оператора SELECT можна використовувати операції порівняння. В результаті буде або істина TRUE, або брехня FALSE. Можна використовувати такі оператори: =, <,>,> =, <=, <>,! <(Не менш),!> (Максимум),! = (Не дорівнює). Пріоритет операції задається круглими дужками.

2.4 Системні функції

Системні функції призначені для отримання інформації про базу даних та її вміст. У SQL сервері існують такі системні функції:

COL_LENGTH (таблиця, поле) – виводить ширину поля;

DATALENGTH (вираз) – виводить довжину вираження;

GETAHSINULL (ім'я БД) – виводить допустимо або неприпустимо використовувати в БД значення NULL;

IDENTINCR (таблиця) – виводить крок збільшення поля лічильника в таблиці;

IDENT_SEED (таблиця) – виводить початкове значення лічильників в таблиці;

ISDATE (вираз) – виводить одиницю, якщо вираз є датою і нуль, якщо не є;

ISNUMERIC (вираз) – виводить одиницю, якщо вираз є числовим і нуль, якщо не числовим;

NULIFF (вираз1, вираз2) – виводить нуль якщо вираз1 рівний вирозу2.

2.5 Агрегатні функції

Агрегатні функції – дозволяють обчислювати підсумкові значення по полях таблиці.

AVGL (поле) – виводить середнє значення поля;

COUNT (*)– виводить кількість записів в таблиці;

COUNT (поле) – виводить кількість всіх значень поля;

MAX (поле) – виводить максимальне значення поля;

MIN (поле) – виводить мінімальне значення поля;

STDEV (поле) – виводить середньо квадратичне відхилення всіх значень поля;

STDEVP (поле) – виводить середньоквадратичне відхилення різних значень поля;

SUM (поле) – підсумовує всі значення поля;

TOP n [Percent] – виводить n перших записів з таблиця, або n% записів з таблиці;

VAR (поле) – виводить дисперсію всіх значень поля;

VARP (поле) – виводить дисперсію всіх різних значень поля.

Приклади використання агрегатних функцій:

`SELECT AVG (вік) FROM Студенти` – виводить середній вік студента з таблиці «Студенти».

`SELECT COUNT (ПІБ) FROM Студенти` – виводить кількість різних ПІБ з таблиці «Студенти».

`SELECT Top 100 FROM Студенти` – виводить перші 100 студентів з таблиці «Студенти».

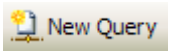
Література: [2, 4].

ЛЕКЦІЯ 5

СТВОРЕННЯ ДИНАМІЧНИХ ЗАПИТІВ ЗА ДОПОМОГОЮ ЗБЕРЕЖЕНИХ ПРОЦЕДУР

Процедура – SQL запит, який має параметри, тобто він виконується як звичайна процедура (ми задаємо її ім'я і передаємо в збережену процедуру значення параметрів.) Залежно від значення параметрів процедури ми отримуємо той чи інший результат запиту.

Зауваження: у SQL сервері збережені процедури реалізують динамічні запити, що виконуються на стороні сервера.

Розглянемо створення збережених процедур за допомогою команд мови SQL. Щоб відобразити збережені процедури робочої БД панелі «Object Explorer» потрібно виділити пункт «Stored Procedures». Щоб створити нову процедуру за допомогою команд мови SQL потрібно клацнути ЛКМ по кнопці 

на панелі інструментів. У робочій області вікна сервера з'явиться вкладка SQLQuery1.sql, де потрібно набрати код нової процедури., Який повинен виглядати так:

```
CREATE PROCEDURE <Ім'я процедури>
[@ <параметр1> <Тип1> [= <значення1>],
@ <параметр2> <Тип2> [= <значення2>], ...]
[WITH ENCRYPTION]
AS <Команди SQL>
```

Тут `Ім'я процедури` – ім'я створюваної процедури;

`Параметр1, параметр2, ...` – параметри які передаються в процедуру;

`Значення1, значення2, ...` – значення параметрів за замовчуванням;

`Тип1, Тип2, ...` – типи даних параметрів;

`WITH ENCRYPTION` – включає шифрування даних;

`Команди SQL` – SQL запит, який виконується при запуску процедур.

Зауваження: SQL запит включає в себе параметри, якщо параметри порівнюються з якими то полями або виразами, то вони повинні мати точно такий же тип даних як ці поля або вирази.

Зауваження: після створення процедура поміщається в розділ Stored Procedures поточної БД на панелі «Object Explorer». Якщо двічі клацнути по процедурі ЛФМ то вона відкриється для редагування на вкладці «SQLQuery».

Щоб подивитися інформацію про збережені процедури необхідно виконати команду: `EXEC SP_HELPTEXT <Ім'я процедури>`

Збережені процедури можуть бути запущені за допомогою такої команди:

```
EXEC <Ім'я процедури> [<параметр1>, <параметр2>, ...]
```

тут `<Ім'я процедури>` – ім'я виконуваної процедури;

`Параметр1, параметр2, ...` – значення параметрів.

Приклад: створення збереженої процедури, який виводить ім'я студентів, у яких середній бал більше заданої величини:

```
 CREATE PROCEDURE СрБАЛЛ
@X Real AS SELECT *
FROM Студенти
WHERE
(Оценка1 + Оценка2 + Оценка3) / 3> @X
```

Команда виклику вищенабранної процедури виглядає наступним чином:

```
EXEC СрБАЛЛ 4
```

Команда виводить всіх студентів, у яких середній бал більше 4.

Література: [2, 4].

ЛЕКЦІЯ 6

КОРИСТУВАЛЬНИЦЬКІ ФУНКЦІЇ

Користувальницькі функції дуже схожі на збережені процедури. Так само в них можна передавати параметри і вони виконують деякі дії, однак їх головною відмінністю від збережених процедур є те, що вони виводять (повертають) якийсь результат. Більш того, вони викликаються тільки за допомогою оператора SELECT, аналогічно вбудованим функціям. Всі призначені для користувача функції діляться на 2вида:

1. **Скалярні функції** – функції, які повертають число або текст, то є одне або кілька значень;

2. **Табличні функції** – функції, які виводять результат у вигляді таблиці.

Для створення нової користувацької функції використовується команда `CREATE FUNCTION` має наступний синтаксис:

```
CREATE FUNCTION <Ім'я функції>
([@<Параметр1> <Тип1>[=<Значення1>],
@<Параметр2> <Тип2>[=<Значення2>], . . .])
RETURNS <Тип>/TABLE
AS
RETURN ([<команди SQL>])
```

Тут `Ім'я функції` – ім'я створюваної користувацької функції;

`Параметр1, Параметр2, ...` – параметри передані в функцію;

`Значення1, Значення2, ...` – значення параметрів за замовчуванням;

`Тип1, Тип2, ...` – типи даних параметрів.

Після службового слова `RETURNS` в скалярних функціях ставиться тип даних результату, який повертає скалярна функція, або ставиться службове слово `TABLE` в табличних функціях.

Після службового слова `RETURN` ставиться SQL команда самої функції.

Зауваження: після службового слова `RETURN` може бути кілька команд, які розташовуються між словами `BEGIN` і `END`. У цьому випадку службове слово `RETURN` не ставиться.

Зауваження: Тип даних параметра повинен збігатися з типом даних виразу, в якому він використовується.

Зауваження: Якщо використовуються кілька SQL команд і `BEGIN` і `END`, то перед `END` потрібно ставити команду `RETURN < результат функції>`.

Приклад (скалярна функція користувача): функція для обчислення середнього з 3 чисел:

```
CREATE FUNCTION Середнє
(@X1_Int,@X2_Int, @ X3_Int)
RETURNS Real
AS
BEGIN
DECLARE_@Res_Real
@Res =(@X1+@x2+@x3)/3
RETURN_@Res
END
```

Зауваження: команда `DECLARE` створює змінну `Res` для зберігання дробових чисел (тип даних Real).

Представлена вище користувацька функція реалізована за допомогою декількох команд SQL, але її можна реалізувати за допомогою однієї функції наступним чином:

```
CREATE FUNCTION Середнє
(@X1_Int,@X2_Int, @ X3_Int)
RETURNS Real
AS
RETURN (SELECT (@X1 + @x2 + @x3)/3)
```

Створена функція, що обчислює середнє 6, 3 і 3, запускається наступним чином:

```
SELECT Середнє (6, 3, 3)
```

Результат буде 4.

Приклад (таблична користувацька функція): З таблиці Студенти виводяться поля ПІБ, дата народження і стовпець вік, який обчислюється як різниця дат в роках, між датою народження і поточною датою (параметр CurDate).

```
CREATE FUNCTION Вік
(@CurDate Date)
RETURNS TABLE AS
RETURN (SELECT ПІБ, [Дата народження], Вік = DATEDIFF
(yy,[Дата народження], @CurDate)
FROM студенти)
```

Ця функція викликається наступним чином:

```
SELECT вік ('12/17/2017')
```

В результаті відобразяться студенти з їх віком на 17 грудня 2017 року.

Література: [2, 4].

ЛЕКЦІЯ 7

ЦІЛІСНІСТЬ ДАНИХ. ДІАГРАМИ І ТРИГЕРИ

1. Цілісність даних
2. Створення тригерів

1. Цілісність даних

При роботі БД повинна забезпечуватися цілісність даних. Під цілісністю даних розуміють забезпечення цілісності зв'язків між записами в таблицях при видаленні записів з первинних таблиць. Тобто, при видаленні записів з первинних таблиць автоматично повинні видалятися пов'язані з ними записи з вторинних таблиць.

У випадку недотримання цілісності даних з часом в БД накопичиться велика кількість записів у вторинних таблицях пов'язаних з неіснуючими записами в первинних таблицях, що призведе до збоїв у роботі БД і її засмічення невикористовуваними даними.

Для забезпечення цілісності даних в SQL Server використовують діаграми і тригери.

Діаграми – це компоненти БД, які блокують видалення записів з первинних таблиць якщо існують пов'язані з ними записи у вторинних таблицях. Отже, діаграми запобігають порушенню цілісності даних. У SQL Server діаграми створюються за допомогою майстра діаграм, його опис представлено в лабораторній роботі.

Тригери – це аналог процедур обробників подій у Visual Basic. Тобто вони виконують команди SQL якщо відбуваються будь-які дії з таблицею (наприклад: додавання, зміна або видалення записів). За допомогою тригерів можна організувати автоматичне видалення записів з вторинної таблиці при видаленні пов'язаної з ними записи з первинної таблиці.

Розглянемо створення тригерів за допомогою мови SQL.

2. Створення тригерів

У SQL Server існують два види тригерів:

1. Тригери виконувані після події, що сталася з таблицею (повний аналог процедур подій у Visual Basic);

2. Тригери виконуються замість події, що відбувається з таблицею. У цьому випадку подія (додавання, зміна або видалення записів) не виконується, а замість нього виконуються SQL команди задані всередині тригера.

Перший вид тригерів застосовується для обробки подій таблиць, а другий – для забезпечення цілісності даних, тобто видалення записів з вторинної таблиці при видаленні пов'язаної з ними запису із первинної таблиці.

Зауваження: тригери створюються для конкретної таблиці і виконуються автоматично якщо з таблиці, для якої вони були створені відбувається подія (додавання, зміна або видалення записів).

Для створення тригера на вкладці нового запиту необхідно набрати команду `CREATE TRIGGER`, що має наступний синтаксис:

```
CREATE TRIGGER <Ім'я тригера>
ON <Ім'я таблиці>
FOR < AFTER|INSTEAD OF> <INSERT|UPDATE|DELETE>
[WITH ENCRYPTION]
AS <Команди SQL>
```

Тут `Ім'я тригера` – це ім'я створюваного тригера;

`Ім'я таблиці` – ім'я таблиці, для якої створюється тригер.

Якщо використовується параметр `ALTER`, то тригер виконується після події, а якщо параметр `INSTEAD OF`, то виконується замість події.

Параметри `INSERT`, `UPDATE` і `DELETE` визначають подію, при якій (або замість якої) виконується тригер.

Параметр `WITH ENCRYPTION` – призначений для включення шифрування даних при виконанні тригера.

`Команди SQL` – це SQL команди, що виконуються при активізації тригера.

Розглянемо приклади створення різних тригерів для таблиці «Студенти».

Приклад: створює тригер «Додавання», що виводить на екран з повідомлення «Запис додано» при додаванні нового запису в таблицю «Студенти»

```
CREATE TRIGGER Додавання
ON Студенти
FOR AFTER INSERT
AS PRINT 'Запис додано'
```

Приклад: створює тригер «Зміна», що виводить на екран повідомлення «Запис змінено» при зміні запису в таблиці «Студенти»

```
CREATE TRIGGER Зміна
ON Студенти
FOR AFTER UPDATE
AS PRINT 'Запис змінено'
```

Приклад: створює тригер «Видалення», що виводить на екран повідомлення «Запис видалено» при видаленні запису з таблиці «Студенти»

```
CREATE TRIGGER Видалення
ON Студенти
FOR AFTER DELETE
AS PRINT 'Запис видалено'
```

Приклад: у цьому прикладі замість видалення студента з таблиці «Студенти» виконується код між `BEGIN` і `END`. Він складається з двох команд `DELETE`. Перша команда видаляє всі записи з таблиці «Оцінки», які пов'язані з записами з таблиці «Студенти». Тобто у яких Оцінки.[Код студента] дорівнює коду студента, що видаляється. Потім з таблиці «Студенти» видаляється сам студент.

```
CREATE TRIGGER ВидаленняСтудента
ON Студенти
INSTEAD OF DELETE
AS
BEGIN
DELETE Оцінки
FROM deleted
WHERE deleted.[Код студента]=Оцінки.[Код студента]
DELETE Студенти
FROM deleted
WHERE deleted.[Код студента]=Студенти.[Код студента]
END
```

Зауваження: Тут видаляємий запис позначається службовим словом deleted.

Зауваження: Для забезпечення цілісності даних тригери використовуються зазвичай разом з діаграмами, але ми можемо застосовувати такі тригери і без діаграм, однак ми не можемо застосовувати діаграми без тригерів.

Література: [2, 4].

ЛЕКЦІЯ 8

ЗАГАЛЬНА ХАРАКТЕРИСТИКА МОВИ VISUAL BASIC 2012. ІСТОРІЯ СТВОРЕННЯ ТА СИСТЕМНІ ВИМОГИ. ОБ'ЄКТИ ЗВ'ЯЗКУ. МАЙСТЕР ПІДКЛЮЧЕНЬ

1. Загальна характеристика мови. Історія створення та системні вимоги
2. Об'єкти зв'язку
3. Майстер підключень
4. Настройка зв'язку підключення вручну

1. Загальна характеристика мови. Історія створення та системні вимоги

Мова програмування Visual Basic 2012 входить до складу пакета Microsoft Visual Studio 2012. Він дозволяє створювати додатки для ОС Windows 2000, XP, VISTA і ОС Windows Mobile, Windows Pocket PC.

Microsoft Visual Basic 2012 володіє наступними особливостями:

1. Для роботи програм, написаних цією мовою, необхідно щоб була встановлена бібліотека Microsoft Net. Frame Work 2.0.

2. Можливість створювати різні частини проекту на різних мовах програмування, що входять в Visual Studio.

3. Можливість використання нових візуальних ефектів доступних Windows XP.

4. Можливість конвертації проектів Visual Basic більш ранніх версій.

5. Велика орієнтація на мережеві технології.

6. Більш спрощена робота з БД. Орієнтація на мову форматування XML. До складу Visual Studio входить SQL Server Express – урізана клієнтська версія SQL Server 2012.

7. Автоматичне підключення всіх доступних компонентів.

Порівняно з Visual Basic6.0, Visual Basic 5.0 і Visual Basic2008, Visual Basic 2012 володіє великими системними вимогами. Для його роботи необхідний комп'ютер, що має наступну конфігурацію:

– процесор Pentium 600 МГерц і вище;

– 256 Мб пам'яті;

– для установки тільки Visual Basic необхідний 1Гб вільного місця на диску для установки пакета Visual Studio необхідно 4Гб .

Visual Basic 2012 базується на ядрі Visual Basic 8.0, який входить до складу Visual Studio 8.0. І був створений в 2008. Після створення Visual Studio 8.0, він отримав велике поширення в світі. У 2003р була створена нова версія Visual Basic Net.

Його головною відмінністю була велика орієнтація на компоненти мережі, використання бібліотеки Microsoft Frame Work 1.0. покращилися графічні спецефекти програми. В 2005 році на основі Visual Basic Net створюється Visual Basic 2005, а в 2008 році створюється поліпшена версія мови Visual Basic 2008.

Створення інтерфейсу клієнтського додатка в Visual Basic відбувається в кілька етапів:

– створюється проект;

– у проекті створюються об'єкти зв'язку, які підключаються до файлу даних;

– створюються форми;

– створюються звіти.

2. Об'єкти зв'язку

Об'єкти зв'язку – це об'єкти проекту, що здійснюють обмін інформацією між інтерфейсом БД і файлом даних.

Об'єкти зв'язку завжди знаходяться на клієнтській машині. Вони здійснюють доступ до файлів даних, передаючи інформацію в інтерфейс БД, і містять всередині себе запити, виконанні на стороні клієнта.

Зауваження: Об'єкти зв'язку також можуть обмежувати доступ до інформації і здійснювати захист інформації, хоча для захисту інформації та обмеження доступу краще використовувати сам сервер.

Існує три технології використовуваних в об'єктах зв'язку:

– технологія ADO;

– технологія RDC;

– технологія ADO.Net.

ADO є більш старою технологією. Її суть полягає в наступному: підключення до конкретної таблиці або запиту здійснюється через окремий об'єкт зв'язку, тобто всі налаштування і засоби для роботи з даними зберігаються всередині конкретного об'єкта зв'язку і були закладені туди при його проектуванні.

Згідно технології RDC файли даних розглядаються в якості пристроїв, тобто для робіт з БД нам необхідний драйвер. Об'єкт зв'язку, що працює за технологією RDC, при роботі з файлом даних спочатку звертається до драйвера БД, який в свою чергу звертається до файлу даних.

Технологія ADO.Net є сумішшю технологій ADO і RDC. Об'єкти зв'язку працюють за цією технологією, працюють аналогічно об'єктам які працюють за технологією ADO, однак, об'єкти зв'язку входять до складу пакета Microsoft Net Framework, і автоматично оновлюються разом з цим пакетом.

Плюси і мінуси технології ADO:

+ незалежність від драйверів БД, встановлених в операційній системі;

+ просте програмування;

- неможливість працювати з новими типами БД;

- неможливість оновлювати список підтримуваних БД.

Плюси і мінуси технології RDC:

+ можливість працювати з сучасними БД;

+ можливість додавати нові види БД;

- залежність від драйверів, встановлених в системі;

- більш складне програмування.

Плюси і мінуси технології ADO.Net:

+ можливість працювати з сучасними БД;

+ можливість додавати нові види БД;

- залежність від пакету Microsoft Net Framework;

- більш складне програмування.

Зауваження: ми можемо створювати динамічні запити, виконані на стороні сервера тільки в технології RDC і ADO.Net.

3. Майстер підключень

В Visual Basic 2012 порівняно з Visual Basic 6.0 підключення проекту до файлу БД можна зробити двома способами: за допомогою майстра підключень і вручну, створюючи об’єкти зв’язку та налаштовуючи їх властивості. Почнемо розгляд створення підключення за допомогою майстра.

Як говорилося вище, об’єкти зв’язку забезпечують доступ до файлів даних. Створення підключення складається зі створення наступних об’єктів:

– DataSet (Набір даних) – забезпечує підключення форми до конкретної БД на сервері (в нашому випадку це БД Students);

– BindingSource (Джерело зв’язку) – забезпечує підключення до конкретної таблиці (в нашому випадку до таблиці «Спеціальності»), а також дозволяє управляти таблицею;

– TableAdapter (Адаптер таблиць) – забезпечує передачу даних з форми в таблицю і навпаки.

– TableAdapterManager (Менеджер адаптера таблиць) – управляє роботою об’єкта TableAdapter;

– BindingNavigator (Панель керування таблицею) – блакитна панель з кнопками управління таблицею, розташована у верхній частині форми.

Можна створити і підключити всі ці об’єкти вручну, але зручніше скористатися майстром. Робота з майстром підключень складається з декількох етапів:

1) Запуск майстра;

2) Вибір типу джерела даних: БД, мережевий джерело або об’єкт;

3) Налаштування рядка підключення «Connection String». Налаштування полягає у виборі виду БД (або Access або SQL Server), а також у виборі сервера і файлу даних. У разі необхідності можна задати логін і пароль;

4) Збереження рядка підключення. При її збереженні можна змінювати параметри підключення без використання Visual Basic. Але при збереженні рядка підключення в файл велика ймовірність несанкціонованого підключення до БД;

5) Вибір таблиць або запитів включених в з’єднання. Також можна вибрати їх окремі поля;

6) Завершення роботи майстра підключень.

Більш докладні інструкції по роботі з майстром підключень можна знайти в лабораторній роботі.

Зауваження: після закінчення роботи майстра підключень. У браузері в «Solution Explorer з’явиться додатковий файл набору даних з розширенням xsd. Цей файл містить у собі схему даних із джерела даних, а також дозволяє редагувати джерело даних (при відкритті цього файлу з’являється вікно схоже на конструктор запитів в Access або SQL Server), в цьому вікні можна також редагувати поля таблиць.

Зауваження: в одному проекті може бути кілька наборів даних, тобто можна запускати майстер підключень скільки завгодно разів. Нові набори даних додаються на вкладку «Data Sources» і з’являється нові дані з розширенням xsd.

4. Настройка зв’язку підключення вручну

В Visual Basic 2012, як і в Visual Basic 6.0 ми можемо створювати об’єкти зв’язку вручну і їх налаштовувати. Для зв’язку Visual Basic 2012 використовує три об’єкти зв’язку, причому вони працюють всі разом, плюс до цього існує об’єкт BindingNavigatior (Панель навігації) – ця панель забезпечує повне управління джерелом даних (додавання, видалення, переміщення по записах).

Розглянемо створення і налаштування відповідних об’єктів зв’язку в порядку черговості:

1) Створення підключення починається з створення об’єкта DataSet. Об’єкт DataSet не може сам підключитися до джерела даних перед його створенням необхідно налаштувати «Data Sources» (віконне меню Data\Add Data Sources). Після створення об’єкта DataSet з’являється вікно «Add DataSet». У ньому

необхідно в випадаючому списку «Typed DatSet» вибрати джерело даних з «Data Sources». Фактично «DataSet» аналогічний Connection з Visual Basic 6.0. Після вибору джерела даних у списку «Typed DataSet» з'явиться рядок Windows Application <Ім'я джерела>. Після цього у вікні можна натиснути кнопку «Ok». Ім'я джерела даних буде записано в властивість datasetname об'єкта DataSet.

2) Після створення об'єкта DataSet створюється об'єкт BindingSource. Цей об'єкт відіграє ту ж роль, що і Command в Visual Basic 6.0, він дозволяє підключиться до таблиць, запитів і фільтрів з файлу даних. Після його створення необхідно налаштувати наступні властивості:

– DataSource – зазначений об'єкт DataSet;

– DataMember – вказує таблицю, запит або фільтр, які будуть відображатися на формі.

Наступні властивості необов'язкові для налаштування:

– Filter – властивість для фільтрації даних, в ньому записується умова відбору для якогось поля;

– Sort – сортування інформації

– Allow New – дозволяє додавати нові записи.

3) Після додавання DataSet і Bindingsource автоматично буде додано Об'єкт TableAdaper. Після чого вже можна додавати об'єкти для відображення даних, однак, при цьому не можна буде управляти інформацією.

4) Для управління джерелом даних створюється об'єкт BindingNavigatior. Потім його необхідно підключити до об'єкта BindingSource. Для цього у властивості BindingSource об'єкта BindingNavigatior необхідно вказати створений раніше об'єкт BindingSource.

Потім можна налаштувати зовнішній вигляд панелі навігації за допомогою наступних властивостей:

AddNewItem – відображає кнопку для додавання нового запису;

DeleteItem – відображає кнопку для видалення поточного запису;

AddNextItem – відображає кнопку для додавання нового запису після поточного;

MoveFirstItem – відображає кнопку для переходу до першого запису;

MoveNextItem – відображає кнопку для переходу до наступного запису;

MovePreviousItem – відображає кнопку для переходу до попереднього запису;

MoveLastItem – відображає кнопку для переходу до останнього запису;

CountItem – відображає загальну кількість записів;

Position Item – відображає номер поточного запису.

Література: [1, 3].

ЛЕКЦІЯ 9

ІНТЕРФЕЙС ІНФОРМАЦІЙНИХ СИСТЕМ. СТВОРЕННЯ ІНТЕРФЕЙСУ КОРИСТУВАЧА

1. Інтерфейс інформаційних систем
2. Створення інтерфейсу користувача за допомогою вікна «Data Sources»
3. Підключення об’єктів до джерела даних за допомогою вікна властивостей

1. Інтерфейс інформаційних систем

В системах, побудованих за технологією клієнт-сервер існує два види інтерфейсу:

– інтерфейс, реалізований за допомогою клієнтського додатка;

– web-інтерфейс.

Інтерфейс, реалізований за допомогою клієнтського додатка – це комп’ютерна програма, що встановлюється на клієнтські комп’ютери, призначена для роботи з файлами даних через мережу. Основними елементами клієнтських додатків є форми (вікно програми) і звіти.

Елементи управління на формі називаються об’єктами. Кожен об’єкт володіє своїм набором властивостей, подій і методів.

Властивості об’єкта – це його характеристики (висота, ширина і інше).

Події об’єкта – це події операційних систем або події які ініціюються користувачем, на які може реагувати об’єкт (натискання кнопки).

Методи об’єкта – дії, які можна проводити з об’єктом в ході виконання програм.

У БД всі об’єкти форм діляться на два класи:

– **об’єкти управління** – об’єкти, що здійснюють управління БД (Наприклад: Кнопка або Випадаючий список);

– **об’єкти для відображення інформації** – елементи, що відображають вміст таблиць, запитів або фільтрів, які дозволяють додавати, змінювати і видаляти інформацію, і проводити її аналіз.

Всі форми в клієнтському додатку діляться на три групи:

1. **Форми для роботи з даними** – форми, що містять як об’єкти управління, так і об’єкти перегляду даних. Такі форми призначені для відображення, зміни, видалення та аналізу даних;

2. **Кнопкові форми** – форми, що містять тільки об’єкти управління, призначаються для відкриття всіх інших форм.

Зауваження: кнопкова форма, яка з'являється першою після запуску програми, називається, головною кнопковою формою.

3. **Інформаційні та службові форми** – форми, що містять елементи керування, призначені для відображення службової інформації (довідки), незв'язаної з таблицями, запитами і фільтрами, або для виконання службових операцій не пов'язаних з даними (Наприклад: форма з калькулятором)

Зауваження: існує два види дизайну форм:

– стрічкові форми – форми, що виводять інформацію по одному запису;

– табличні форми – форми виводять інформацію у вигляді таблиці.

Зауваження: об'єкти зв'язку використовуються тільки в клієнтському інтерфейсі. У web-інтерфейсі функції об'єкта зв'язку виконує сервер.

Основою web-інтерфейсу є сторінки (файл з розширеним htm або html). Робота зі сторінками здійснюється за допомогою програми – браузера. Спочатку сторінки знаходяться на сервері, користувач спочатку завантажує їх на свій комп'ютер з сервера, а потім за допомогою сторінок користувач працює з файлом даних.

2. Створення інтерфейсу користувача за допомогою вікна «Data Sources»

Visual Basic 2012 дозволяє створювати нескладний інтерфейс БД, без допомоги панелі об'єктів і вікна властивостей, лише використовуючи вікно «Data Sources». У вікні «Data Sources» після підключення джерела даних відображаються всі таблиці, запити, фільтри даних і їх поля. В Visual Basic 2012 як і в Visual Basic 6.0 можна перетягувати джерела даних, відповідно таблиці, запиту, фільтру прямо з вікна «Data Sources» на форму (у Visual Basic 6.0 вікно «Data Sources» замінено на вікно «Data Environment»). Головна відмінність Visual Basic 2012 є те, що при перетягуванні можна вибирати для кожного поля джерела даних об'єкт, який буде відображати його вміст.

Зауваження: таким способом можна створювати лише певні об'єкти для відображення даних поля, і набір цих об'єктів залежить від типу даних поля.

Створення об'єктів для відображення даних перетягуванням складається з двох кроків:

1. Для кожного поля таблиці, запиту або фільтру вибирається об'єкт, який буде відображати його вміст. Для цього необхідно клацнути мишею по полю у

вікні «Data Sources», поруч з ім’ям поля з’явиться кнопка зі стрілкою, клацнувши мишею по стрілці, з’явиться випадаюче меню з об’єктами, які можуть відображати інформацію, що міститься в полі. Для полів стандартними об’єктами є: TextBox, ComboBox, Label, LinkLabel, ListBox, LinkLabel. Для полів типу даних дата час (DateTime) можливе використання об’єкта DataTimePicker. Для полів логічних типів даних можливе використання об’єкта CheckBox. Для відображення таблиць, запитів або фільтрів цілком можливо два варіанти відображення: 1) За допомогою об’єкта DataGridView – інформація з таблиці, запиту або фільтру відображається у вигляді таблиці; 2) DetalledView – відображення всіх полів джерела даних в TextBox окремо

Зауваження: у випадаючому меню з варіантами вибору об’єктів є пункт «Customize» (Налаштування), який дозволяє вибрати додаткові допустимі об’єкти для відображення інформації.

2. Після вибору об’єктів для відображення їх необхідно помістити на форму, перетягуючи мишею з панелі «Data Sources» в потрібне місце на формі.

Зауваження: при приміщенні першого об’єкта на форму на ній автоматично створюються об’єкти для зв’язків з файлом даних і об’єкти по навігаціях по джерелах даних (DataSet, BindingSource, TableAdapter, BindingNavigator).

Зауваження: за замовчуванням панель навігації розташовується у верхній частині форми. Цю панель можна прикріпити близько різних країв форми. Для цього необхідно скористатися меню дій об’єктів. Це меню схоже з вікном «Property Pages» з Visual Basic 6.0, але крім основних властивостей об’єктів воно містить і дії, які можна проводити з об’єктами. Щоб викликати це меню необхідно виділити об’єкт. У його правому верхньому куті з’явиться (квадратик зі стрілочкою) при натисканні цієї кнопки з’являється меню, що випадає з настройками і дії з об’єктом. Наприклад, щоб поміняти місце положення навігації панелі треба в цьому меню вибрати настройку Docking.

Зауваження: при перетягуванні на форму полів джерел даних автоматично створюються підписи до них (Label).

Зауваження: після перетягування з створеним об’єктом можна працювати як і зі звичайним об’єктом Visual Basic.

3. Підключення об’єктів до джерела даних за допомогою вікна властивостей

Visual Basic 2012 дозволяє підключати джерела даних до об’єктів без використання перетягування, тобто вручну, з використанням панелі властивостей, як в Visual Basic 6.0. Для цього на форму поміщається об’єкт, який буде підключатися до джерела даних. Його виділяють, потім на панелі властивостей розгортається група властивостей «DataBindings» вона містить дві властивості:

– Text-визначає таблицю, запит або фільтр, з якого виводяться дані в об’єкт;

– Tag-визначає поле, обраного у властивості Text джерела даних, яке відображається в об’єкті.

Література: [1, 3].

ЛЕКЦІЯ 10

СТАНДАРТНІ ОБ’ЄКТИ ДЛЯ ВІДОБРАЖЕННЯ ДАНИХ. ПРОГРАМНЕ УПРАВЛІННЯ ІНФОРМАЦІЙНОЮ СИСТЕМОЮ

1. Стандартні об’єкти для відображення даних
2. Програмне управління інформаційною системою

1. Стандартні об’єкти для відображення даних

Спосіб створення об’єктів для відображення даних описаний раніше, дозволяє створювати тільки обмежений набір об’єктів. Однак, Visual Basic 2012 дозволяє підключати джерела даних практично до будь-якого об’єкту, який може бути створений на формі. Це можна зробити за допомогою перетягування поля джерела даних з вікна «Data Sources» на об’єкт на формі.

Операція складається з двох кроків:

1. За допомогою панелі об’єктів (зліва) на формі створюється якийсь об’єкт. Об’єкти незв’язані з джерелом даних називають незв’язаними об’єктами.

2. Новостворений об’єкт зв’язується з джерелом даних. Для цього на об’єкт потрібно перетягнути поле таблиці запиту або фільтра з вікна «Data Sources».

Зауваження: при перетягуванні поля з вікна «Data Sources» необхідно враховувати його тип даних. Об’єкт на формі повинен підтримувати тип даних поля який підключається до нього.

Зауваження: у разі підключення об’єкта до джерела даних, способом, описаним вище, підпис до об’єкта не створюється автоматично і його треба створювати вручну за допомогою об’єкта Label.

Найбільш часто в БД використовуються наступні об’єкти для відображення інформації:

- текстове поле (TextBox);
- напис (Label);
- напис з посиланням (LinkLabel);
- календар (DataPicker);
- перемикач (CheckBox);
- таблиця (DataGridView);
- список (ListBox);
- список, що випадає (ComboBox);

– текстове поле з маскою введення (MaskedTextBox).

TextBox – відображає текст і числові поля, це найбільш часто вживаємий об’єкт для відображення даних. Його можна створювати або перетягуванням з вікна «Data Sources», або підключити вручну. Створення цього об’єкта, перетягуванням можливо у полів будь-яких типів даних.

Label – повністю аналогічний об’єкту TextBox, але не дозволяє змінити дані. Цей об’єкт використовується для відображення заблокованих незмінних полів.

LinkLabel – спеціальний об’єкт для відображення посилань на адреси в Інтернеті. Його використовують для відображення текстових полів, якщо в них зберігаються адреси Інтернету або якоїсь комп’ютерної мережі. Це новий об’єкт, йому не було аналога в Visual Basic 6.0.

DataPicker – спеціальний об’єкт, призначений для відображення полів типу даних «Дата/Час» у вигляді календаря.

CheckBox – об’єкт використовується для відображення логічних полів, може бути створений перетягуванням тільки для логічних полів.

DataGridVilew – об’єкт, що відображає джерело даних таблицю, запит або фільтр у вигляді таблиці.

ListBox – список відображає значення полів і дозволяє вибирати значення полів зі списку. Більш того, пункти списку можна задавати, використовуючи інше джерело даних.

ComboBox – об’єкт подібний об’єкту LiasBox, проте інформація відображається не в списку, а в випадаючому списку.

MaskedTextBox – нестандартний об’єкт, призначений для відображення і введення інформації за заздалегідь заданим шаблоном (масці). Цей об’єкт може бути створений тільки за допомогою панелі об’єктів та його підключення здійснюється або перетягуванням на нього поля з вікна «Data Sources», або завданням його властивостей вручну. За своїми властивостями він нічим не відрізняється від об’єкта TextBox. Єдина додаткова властивість у цього об’єкта це властивість Mask. Для цього потрібно клацнути по кнопці дій об’єкта у верхньому правому куті об’єкта. Потім в списку дій вибрати пункт «Edit Mask». У вікні вибрати шаблон введення, тобто маску (Mask).

Зауваження: тип даних відображуваної інформації повинен збігатися з типом даних маски.

Зауваження: об’єкти ListBox і ComboBox можуть використовуватися для заповнення полів з кодами, тобто списки заповнюються інформацією з однієї таблиці, а при виборі пункту списку його код підставляється в іншу таблицю.

Для цього на форму мають не підключений ListBox або ComboBox, потім відкриваємо його меню дій. У меню дій у зазначеній послідовності виконують наступні кроки:

– встановити галочку «Use Data-Bound Items»;

– у випадаючому списку «Data Source» вибрати пункт «Other Data Source» і там вибрати потрібну таблицю;

– у випадаючому списку «Display Member» і вказуємо поле, яке відображається в списку;

– у випадаючому списку «Value Member» вказуємо поле, яке підставляємо при виборі пункту списку.

– у випадаючому списку «Selected Value» вказуємо поле, куди підставляється вибране в «Value Member» значення.

2. Програмне управління інформаційною системою

У Visual Studio 2012 додавати, видаляти записи і переміщатися ним можна як використовуючи об’єкт Navigator, так і використовуючи звичайні кнопки. Розглянемо створення кнопок для управління записами. У Visual Studio 2012 відсутній об’єкт «RecordSet» всі операції з записами здійснюються з використанням об’єкта «BindingSource».

В Visual Basic 2012 для додавання нового запису до таблиці «Студенти» використовується команда виду `СтудентыBindingSource.AddNew`. Замість методу AddNew можна використовувати методи:

– MoveNext (перейти до наступної);

– MoveFirst (Перейти до першої);

– Move Previous (Перейти до попередньої);

– Move Last (Перейти до останньої);

– Delete (Видалити запис).

Об’єкт BindingSource має властивість Filter. Воно аналогічно Filter у об’єкта RecordSet в Visual Basic 6.0. Його використання нічим не відрізняється від виконання такої ж властивості в Visual Basic 6.0. Тобто у властивості Filter задається рядок, що визначає умову відбору записів у динамічних фільтрах, що виконуються на стороні клієнта. Цей рядок має наступний синтаксис:

```
<Поле1><Оператор1><Вираз1>
[And|OR <Поле2> <Оператор2> <Вираз2>…]
```

Тут `<Поле1>, <Поле2> . . .` – поля на які накладаються умови;

`<Оператор1>, <Оператори2>` – оператори порівняння, що беруть участь в умовах;

<Вираз1>, <Вираз2> – вирази з якими порівнюються поля. Під виразами розуміються, константи, змінні, формули, функції і властивості об’єктів.

Приклад: з таблиці «Студенти» необхідно відобразити студента, у якого значення поля ПІБ рівне «Петров».

```
СтудентиBindingSource.Filter = «ПІБ = ‘Петров‘»
```

Зазвичай при формуванні запиту за допомогою властивості Filter для завдання умов відбору використовують або списки ListBox, або випадні списки ComboBox.

Зауваження: якщо ми використовуємо ComboBox для створення динамічного фільтра, то в меню дій параметри «Value Member» і «Selected Value» налаштовувати не треба.

Приклад: є таблиця «Студенти», яка відображається на формі в DataGridView. Необхідно на формі помістити ComboBox з прізвищами студентів. При виборі ПІБ і натисканням на кнопку відобразити дані тільки по обраному студенту.

В цьому випадку в меню дій ComboBox в параметрі «Data Source» вказуємо «Other Data Source/Студенти». Потім в «Display Member» вибираємо ПІБ. У коді кнопки прописуємо наступну команду:

```
СтудентиBindingSource.Filter = «ПІБ=‘» & ComboBox1.Text & «‘»
```

Після натискання кнопки в DataGridView відображаються дані по студенту, обраному в випадаючому списку ComboBox1.

Література: [1, 3].

ЛЕКЦИЯ 11.

ОБ’ЄКТ ДЛЯ ВІДОБРАЖЕННЯ ТАБЛИЧНОЇ ІНФОРМАЦІЇ DATAGRIDVIEW. НАЛАШТУВАННЯ ВЛАСТИВОСТЕЙ СТОВПЦІВ В DATAGRIDVIEW

1. Об’єкти для відображення табличної інформації DataGridView
2. Налаштування властивостей стовпців в DataGridView

1. Об’єкт для відображення табличної інформації DataGridView

Об’єкт DataGridView призначений для відображення всієї інформації з таблиць, запитів або фільтрів на формі у вигляді таблиці. Цей об’єкт може бути створений як вручну (з подальшим його підключенням), так і перетягуванням всього джерела даних з вікна «Data Sources». Однак найбільш часто його створюють перетягуванням всієї таблиці, запиту або фільтра з вікна «Data Sources» на форму.

При перетягуванні об’єкта на форму, як і у випадку з іншими об’єктами з’являється панель навігації. Вона виконує функції: переміщення по записах, додавання, видалення і збереження записів. Після створення Об’єкта DataGridView можна настроювати як властивості всього об’єкта, так і властивості окремих стовпців. Почнемо з налаштування властивостей всього об’єкта. Налаштування даних властивостей здійснюється в основному через меню дій. Можливі наступні настройки:

Chose Data Source – джерело даних, що відображається в таблиці;

Enable Adding – додавати записи;

Enable Deleting – дозволяється користувачам видаляти записи;

Enable Editing – дозволяється користувачам змінювати значення полів таблиці;

Enable Column Reordering – дозволяється користувачам змінювати порядок стовпців, перетягуючи їх мишею.

Також в меню дій можливі наступне дії з таблицею:

Dock in parent container – вписати об’єкт у форму;

Preview Data – з’являється вікно з попереднім переглядом таблиці;

Add Query – додає SQL-запит, який виконується на стороні клієнта;

Add Column – додавання нового стовпця у таблицю;

Edit Columns – налаштування властивостей окремих стовпців таблиці.

Тепер перейдемо до налаштування окремих стовпців таблиці.

2. Налаштування властивостей стовпців в DataGridView

Якщо в меню дій вибрати пункт «Edit Columns», то з'являється вікно, де можна додавати, видаляти і редагувати стовпці. Для цього в списку стовпців лівої частини вікна вибираємо стовпець, а в правій – налаштовуємо його властивості.

Найбільш часто налаштовуються такі властивості:

Name – ім'я стовпця;

AutoSizeMode – підгонка ширини стовпця по його вміст;

ColumnType – визначає зовнішній вигляд комірок стовпця (який об'єкт для відображення інформації знаходиться в комірках стовпця);

DataPropertyName – ім'я, що відображає в стовпці поля;

Frozen – фіксація стовпця (стовпець не пересувається при прокручуванні таблиця);

HeaderText – текст заголовка стовпця;

Width – ширина поля;

MaxInputLength – максимально вводима довжина тексту;

MinimumWidth – мінімальна ширина стовпця;

ReadOnly – блокування стовпця для редагування даних;

Resizable – дозволяє змінювати ширину стовпця;

SortMode – сортування даних у таблиці за цим стовпцем;

ToolTipText – підказка для стовпця;

Visible – робить стовпець невидимим.

Зауваження: для додавання нових стовпців у вікні «Edit Columns» необхідно натиснути кнопку Add, а для видалення кнопку Remove.

Зауваження: якщо необхідно налаштувати зовнішній вигляд всіх осередків таблиці, то для цього необхідно виділити об'єкт DataGridView і на панелі властивостей зайти у властивість DefaultCellStyle. З'явиться вікно з властивостями всіх осередків таблиці.

Зауваження: в об'єкті DataGridView є можливість сортування даних. Для цього використовується метод Sort, що має наступний синтаксис:

`DataGridView.Sort (<ім'я стовпця>, <порядок сортування>)`

де `DataGridView` – це ім'я об'єкта, `<Ім'я стовпця>` – це ім'я стовпця (властивість Name) по якому відбувається сортування записів у таблиці, параметр `<Порядок сортування>` визначає порядок сортування і може приймати два значення:

– System.ComponentModel.ListSortDirection.Ascending – сортування за зростанням;

– System.ComponentModel.ListSortDirection.Descending – сортування за спаданням.

Зауваження: доступ до окремих осередків таблиці можна отримати через під об’єкт Item.

Звернення до нього здійснюється наступним чином:

```
DataGridView.Item(i, j).<Властивість>
```

Тут `DataGridView` – це ім’я об’єкта, `i` – горизонтальна координата комірки, а `j` – вертикальна, `<властивість>` – це настроювана властивість комірки.

Приклад: у верхню ліву комірку таблиці записати слово «Привіт» і зробити колір тексту в комірці червоним.

```
DataGridView.Item(0, 0).Value = «Привіт» DataGridView.Item(0, 0).Style.ForeColor = Color.Red
```

Тут `DataGridView` – це ім’я об’єкта, властивість `Value` визначає вміст комірки таблиці, властивість `Style.ForeColor` визначає колір тексту в комірці. Нумерація стовпців і рядків в таблиці починається з нуля.

Література: [1, 3].

ЛЕКЦІЯ 12

ЗВІТ. ОБ'ЄКТИ ДЛЯ РОБОТИ ЗІ ЗВІТАМИ

1. Звіт
2. Об'єкти для роботи зі звітами

1. Звіт

Клієнтські додатки здійснюють виведення інформації на друк за допомогою звітів. Звіти так само, як і форми складаються з об'єктів і самі є об'єктами, але між звітами і формами є відмінності:

1. Звіти містять тільки об'єкти для відображення інформації (наприклад, підписи, малюнки, текстові поля, геометричні фігури і лінії), але не містять об'єкти для управління системою (наприклад, кнопки або випадні списки).

2. У звітах відразу ж виводитися всі записи з джерела даних (таблиці, запиту або фільтра) і висновок виводиться на листи.

3. Звіт можна створити без наявності в системі принтера, так як налаштування зовнішнього вигляду звіту беруться з налаштувань драйвера принтера.

4. На відміну від форм звіти складаються з п'яти розділів:

– заголовок – верхня частина першого листа звіту. У заголовку мають назву звіту і деяку службову інформацію. Наприклад, герб і юридична адреса фірми або ім'я автора звіту.

– примітка – нижня частина останнього листа звіту. У примітку поміщають підсумкову інформацію по звіту (наприклад, загальний обсяг продажів, всіх угод представлених у звіті) і місце для друку і підпису керівника.

– верхній колонтитул – верхня частина кожного аркуша звіту. В даний розділ поміщають номери аркушів звіту і додаткову службову інформацію. Наприклад, дату і час створення звіту.

– нижній колонтитул – нижня частина кожного аркуша звіту. В даному розділі розташовується та ж інформація, що і у верхньому колонтитулі, але не дублює інформацію з верхнього колонтитула.

– область даних – середня частина кожного листа звіту.

Зауваження: існує два види дизайну звітів:

– **стрічковий дизайн** – виводить інформацію по кожному запису окремо. Тобто для кожного поля кожного запису відображається назва поля і його значення;

– **табличний дизайн** – виводить інформацію у вигляді таблиці. Тобто в заголовок звіту поміщають назви полів, а в області даних під назвою полів відображаються їх значення.

2. Об’єкти для роботи зі звітами

Робота зі звітами в Visual Basic2012 складається з декількох етапів:

1. Створюється порожній звіт.
2. У звіт поміщають об’єкти для відображення інформації.
3. Створюється форма для відображення звіту.
4. На форму поміщають об’єкт Reportvewer, що відображає звіти.
5. До об’єкта Reportvewer підключають створений раніше звіт.

Створення звітів і відображання їх форм детально розглянуто в лабораторній роботі. Зупинимося більш докладно на звіті і об’єктах, використовуваних при його створенні.

Для створення порожнього звіту у віконному меню необхідно вибрати пункт «Project\Add New Item...» і у вікні в розділі «Reporting» двічі клацнути ЛКМ по пункту «Report» з’явиться вкладка з порожнім звітом. Тепер у звіт необхідно додати об’єкти для відображення даних.

Зауваження: робота з об’єктами в звіті повністю аналогічна роботі з об’єктами на формі. Тобто ми можемо перетягувати поля в звіт з вікна «Data Sources» або можемо створити об’єкти в звіті вручну, а потім підключити їх до полів через панель властивостей.

Зауваження: у звітах всі об’єкти діляться на об’єкти контейнери, об’єкти для відображення даних і об’єкти оформлення.

– **об’єкти контейнери** – це об’єкти, що містять об’єкти для відображення даних і визначають дизайн звіту;

– **об’єкти для відображення даних** – це об’єкти, що відображають значення полів джерела даних або додаткову службову інформацію;

– **об’єкти оформлення** – об’єкти, що застосовуються тільки для оформлення звіту.

Розглянемо об’єкти для відображення даних, до них відносяться:

– TextBox – текстове поле введення, призначене для відображення значень полів і будь-якої текстової інформації. Якщо об’єкт TextBox використовується для відображення інформації з джерела даних, і він знаходиться поза об’єкту контейнера, то в ньому буде відображено значення вибраного поля тільки першого запису з джерела даних;

– Image – об’єкт відображає вміст полів з графічною інформацією або відображає малюнки (графічні файли);

– Chart – об’єкт, що відображає графік або гістограму побудовану за інформацією з джерела даних.

Тепер розглянемо об’єкти контейнери. У звіт можна помістити такі об’єкти контейнери:

– Table – таблиця виводить інформацію у вигляді таблиці з обмеженою кількістю стовпців і необмеженою кількістю рядків. Тобто кількість рядків у таблиці залежить від обсягу даних які виводяться;

– Matrix – таблиця виводить інформацію у вигляді таблиці з необмеженою кількістю стовпців і рядків. Тобто кількість рядків і стовпців в таблиці залежить від обсягу виведених даних;

– List – об’єкт виводить інформацію у вигляді списків;

– Subreport – об’єкт містить всередині себе додатковий звіт, створений раніше.

Нарешті, розглянемо об’єкти оформлення. До них відносяться:

– Line – відображає лінію;

– Rectangle – відображає прямокутник, використовується для угруповання полів.

Зауваження: робота з рядками, стовпцями або комірками об’єктів Table, Matrix і List здійснюється як у програмі «Microsoft Excel».

Зауваження: у комірках об’єктів Table, Matrix і List можна друкувати текст, як і в комірках таблиць "Microsoft Excel". Щоб помістити в комірку значення поля його можна перетягнути з вікна «Data Sources» в комірку, або в комірці написати код, що має наступний синтаксис: `= Fields!<Ім’я поля>.Value` де ім’я поля – це ім’я відображуваного поля. Аналогічно можна проводити обчислення в осередках.

Приклад: у комірці відобразити середній бал трьох полів: Оцінка1, Оцінка2 і Оцінка3. Для вирішення цього завдання в комірці необхідно набрати код:

```
= (Fields!Оценка1.Value+Fields!Оценка2.Value+
Fields!Оценка3.Value)/3
```

Література: [1, 3, 4].

СПИСОК РЕКОМЕНДОВАНОЇ ЛІТЕРАТУРИ

1. Кузьменко В. Г. Базы данных в Visual Basic и VBA. Самоучитель. – М.: ООО «Бином-Пресс», 2004. – 416с.;

2. Пирогов В. Ю. SQL Server 2005: программирование клиент-серверных приложений. Спб.: БХВ-Петербург, 2006. – 336с.;

3. Стивенс Р. Программирование баз данных. М.: ООО «Бином-Пресс», 2007. – 384с.;

4. Уолтерс Роберт, Коулс Майкл, Рей Роберт, Феррачати Фабио, Дональд Фармер SQL Server 2008. Ускоренный курс для профессионалов, Вильямс – Москва – Санкт Петербург – Киев, 2008 – 768с.