

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

МЕТОДИЧНІ ВКАЗІВКИ
до самостійної роботи (мова SQL)
з дисципліни «Системи керування базами даних»
для студентів спеціальності
151 – «Автоматизація та комп'ютерно – інтегровані технології»
всіх форм навчання

2020

Методичні вказівки до самостійної роботи (мова SQL) з дисципліни «Системи керування базами даних» для студентів спеціальностей 151 – «Автоматизація та комп'ютерно – інтегровані технології» всіх форм навчання ./Укл. доц. С.І. Арсеньєва, – Запоріжжя: НУ Запорізька політехніка, 2020.-28 с.

Укладач: С.І.Арсеньєва, доцент, к.ф-м.н. .

Рецензент: В.В.Корольков, доцент, к.т.н.

Відповідальний за випуск: зав. лаб. каф. ЕПА К.І.Пилипенко.

Затверджено на засіданні кафедри
“Електропривод та автоматизація
промислових установок”
Протокол № 12 від 12.05.2020 р.
Рекомендовано до
видання НМКЕТФ
протокол № 8
від 18 червня 2020 р.

ЗМІСТ

ЛАБОРАТОРНА РОБОТА № 1	4
СТВОРЕННЯ SQL ЗАПИТІВ	4
1 ЗАГАЛЬНІ ВІДОМОСТІ.....	4
Побудова простих запитів за допомогою оператора вибору SELECT	4
Синтаксис оператора SELECT:	4
Пропозиція WHERE	5
Пропозиція ORDER BY	8
Пропозиція Group by	9
Застосування агрегатних (узагальнюючих) функцій	9
Пропозиція HAVING	11
Вкладені запити.....	12
Використання вкладеного запиту з узагальнюючими функціями	13
Використання предиката IN.	13
Використання предиката EXIST.	13
Використання вкладеного запиту з перевіркою рівності	14
Багатотабличні запити	14
Синтаксис з'єднаних таблиць	14
2 ЛАБОРАТОРНЕ ЗАВДАННЯ.....	17
3 ЗМІСТ ЗВІТУ.....	21
4 КОНТРОЛЬНІ ПИТАННЯ.....	21
ЛАБОРАТОРНА РОБОТА № 2	22
СТВОРЕННЯ КНОПКОВОЇ ФОРМИ.....	22
1 ЗАГАЛЬНІ ВІДОМОСТІ.....	22
Кнопкова форма	22
Створення кнопкової форми за допомогою диспетчера кнопок форм.....	22
Оформлення Меню для роботи з базою	24
Стискування бази даних при виході з додатка.....	24
2 ЛАБОРАТОРНЕ ЗАВДАННЯ	25
3 ЗМІСТ ЗВІТУ	276
4 КОНТРОЛЬНІ ПИТАННЯ	27
РЕКОМЕНДОВАНА ЛІТЕРАТУРА	2

Лабораторна робота № 1

Створення SQL запитів

Мета роботи – освоїти основні способи створення запитів з використанням мови SQL.

1 ЗАГАЛЬНІ ВІДОМОСТІ

Запити відносно змісту бази даних та команди та команди зміни бази даних описуються засобами мови SQL.

Основу мови SQL складають оператори, умовно розбиті на кілька груп відповідно до виконуваних функцій:

- Оператори DDL (Data Definition Language) - оператори визначення об'єктів бази даних, наприклад,
- Оператори DML (Data Manipulation Language) - оператори маніпулювання даними, наприклад:
- Оператори захисту і керування даними, наприклад:

Побудова простих запитів за допомогою оператора вибору **SELECT**

Призначення оператора **SELECT** складається у відбиранні і відображенні даних однієї або більш таблиць бази даних. Цей оператор забезпечує виконання дій, еквівалентних операторам реляційної алгебри: вибірка, об'єднання, проекція.

Результатом виконання оператора **SELECT** завжди є таблиця.

Синтаксис оператора *SELECT*:

Послідовність ключових слів в операторі **SELECT** не може бути змінений

SELECT [ALL | DISTINCT] { * [список полів [AS нове ім'я]] [,.....] } *

FROM ім'я таблиці [alias] [,...]

[**WHERE** <предикат-умова вибірки або з'єднання>]

[**GROUP BY** <Список полів>]

[**HAVING** <предикат-умова для групи>]

[**ORDER BY** <Список полів, по яких упорядкувати висновок>]

Квадратними дужками позначені не обов'язкові пропозиції оператору

SELECT.

Оператор SELECT закінчується символом „;”.

SELECT — ключове слово, що повідомляє СУБД, що ця команда - запит. Усі запити починаються с цього слова з наступним пробілом. За ним може зазначатись спосіб вибірки — з видаленням дублікатів (DISTINCT) або без видалення (ALL, за замовчуванням). Потім вказуються імена полів, мають включатись в результуюче відношення. **SELECT** [ALL | DISTINCT] { * [список полів [AS нове ім'я]][,.....]|*)

Пропозиція **SELECT** може включати тільки наступні типи елементів:

- імена стовпців;
- агрегатні функції;
- константи;
- вирази, що включають у себе комбінації перерахованих вище елементів.

Символ * (зірочка) означає, що у результуючий набір включаються всі стовпці з вихідних таблиць запиту.

У пропозиції **FROM** задається перелік вихідних таблиць запиту. **FROM** <Список таблиць> через кому.

FROM - ключове слово, подібно **SELECT**, що повинне бути представлене в кожному запиті. Воно супроводжується пробілом і потім іменами таблиць, використовуваних як джерело інформації. У випадку, якщо зазначено більш одного імені таблиці, над перерахованими таблицями здійснюється операція декартового добутку.

Приклад Створити список студентів, із зазначенням їхній ФІО і групи

```
SELECT ФІО, Група
FROM Студенти;
```

Пропозиція WHERE

Для обмеження набору рядків, що поміщаються в результуючу таблицю запиту використовується пропозиція **WHERE**, де задаються умови добору рядків результату або умови з'єднання рядків вихідних

таблиць, тобто задаються обмеження набору рядків, що поміщаються в результуючу таблицю.

WHERE <Предикат-умова вибірки або з'єднання>

Існує п'ять основних типів умов пошуку (предикатів)

- **Порівняння** – порівнюються результати обчислень двох виразів

Предикати порівняння { =, <, >, <=, >= }, що мають традиційний сенс

Приклад Вибрати всіх студентів, що навчаються в групі E-316

Select Прізвище

From Студенти

Where курс=4;

- **Діапазон** - перевіряється, чи попадає результат обчислення в заданий діапазон значень

Предикат **Between A and Y** — приймає значення між A и B. Коли порівнюване значення попадає в заданий діапазон, включаючи границі діапазону, тоді відповідний рядок включається в результуючу таблицю. Одночасно використовується протилежний предикат **Not Between A and Y**, коли порівнюване значення не попадає в заданий інтервал, включаючи його границі, відповідний рядок не включається у результуючу таблицю.

Приклад: Скласти список студентів з другого по четвертий курси.

SELECT Студенти.*

FROM Студенти

WHERE Студенти.[курс] Between 2 And 4;

- **Приналежність до множини** - перевіряється, чи належить результат до заданої безлічі значень

Предикат входження в множину **IN (множина) вірний** тоді, коли порівнюване значення входить у множину заданих значень. Одночасно існує протилежний предикат **NOT IN (множина)**, що щира тоді, коли порівнюване значення не входить у задану безліч.

Приклад: Скласти список викладачів - доцентів

```
SELECT Викладачі.*
FROM Викладачі
WHERE Посада IN ("Доцент");
```

- Відповідність шаблону – перевіряється, чи відповідає деяке строкове значення заданому шаблону - предикати порівняння зі зразком **LIKE** і **NOT LIKE**. Предикат **LIKE** вимагає завдання шаблону, з яким порівнюється задане значення, предикат вірний, якщо порівнюване значення відповідає шаблону, і помилковий у протилежному випадку. Предикат **NOT LIKE** має протилежний сенс.

По стандарті в шаблон можуть бути включені спеціальні символи:

- символ підкреслення () — для позначення будь-якого одиночного символу;
- символ відсотка (%) — для позначення будь-якої довільної послідовності символів;
- інші символи, задані в шаблоні, позначають самих себе.

Приклад: Список усіх співробітників, які мешкають на вулиці Горького

```
SELECT Викладачі.*
FROM Викладачі
WHERE Адреса LIKE "%Горького%";
```

— адреса, що містить набір символів Горького (при цьому припустимо використання символу * для позначення будь-якої кількості символів)

- Значення NULL- перевіряється, чи містить заданий стовпець визначник NULL

Предикат порівняння з невизначеним значенням IS NULL.

Якщо в даному кортежі (у даному рядку) зазначений атрибут має невизначене значення, то предикат IS NULL приймає значення «Істина» (TRUE), а предикат IS NOT NULL — «Неправда» (FALSE), у протилежному випадку предикат IS NULL приймає значення «Неправда», а предикат IS NOT NULL приймає значення «Істина».

Приклад: Список усіх співробітників, що мають телефони

```
SELECT Викладачі.*
FROM Викладачі
```

WHERE Телефон IS NOT NULL;

• Предикати існування **EXIST** і не існування **NOT EXIST**. Ці предикати відносяться до убудованих підзапитів,

Для побудови більш складних запитів предикати можуть бути побудовані з використанням логічних операторів AND, OR, NOT.

Обчислення у виразах виконуються за наступними правилами:

- вираз обчислюється ліворуч праворуч;
- першими обчислюються вираження в дужках;
- оператори NOT обчислюються до виконання операторів AND або OR;
- оператори AND обчислюються до виконання операторів OR;.
- для усунення яких-небудь неоднозначностей рекомендується використовувати дужки.

Приклад: Перелічити усіх викладачів – доцентів кафедри Фізики
SELECT Викладачі.*

FROM Викладачі

WHERE Кафедра= «Фізики» and Посада = «Доцент»;

Пропозиція ORDER BY

Сортування результатів **ORDER BY** <Список полів> використовується у разі необхідності упорядкування рядків результуючої таблиці, тому що у загальному випадку результати запитів не упорядковані. За замовчуванням сортування встановлене по зростанню **ASC** для сортування по зменшенню **DESC**.

Приклад: Упорядкування результатів запиту по декількох полях зі зростанням або убуттям (ключові слова ASC, DESC):

У результаті одержимо таблицю, у якій рядки йдуть у порядку зростання за значення поля DNUM, а рядки, з однаковим значенням DNUM йдуть у порядку убуття значення поля VOLUME

SELECT PD.PNUM, PD.DNUM, PD.VOLUME

FROM PD

ORDER BY DNUM ASC, VOLUME DESC;

Пропозиція Group by**GROUP BY** <Список полів>

При відображенні таблиці групуються рядки, що мають однакове значення заданого стовпця. Це пропозиція найчастіше використовується з агрегатними функціями.

Приклад:

```
SELECT Студенти.Група, Count(Студенти.Прізвище) AS Прі-
звище
FROM Студенти
GROUP BY Студенти.Група;
```

Застосування агрегатних (узагальнюючих) функцій

У **SQL** додані додаткові функції, що дозволяють обчислювати узагальнені групові значення. Для застосування агрегатних функцій передбачається попередня операція угруповання.. Можна застосовувати агрегатні функції також і без операції попереднього угруповання, у цьому випадку усе відношення розглядається як одна група і для цієї групи можна обчислити одне значення на групу. Агрегатні функції повертають **одне** значення для всієї групи.

Стандарт ISO передбачає 5 узагальнюючих (агрегатних) функцій:

COUNT - підрахунок кількості рядків або не порожніх значень полів, що вибрав запит. Застосовується для числових і нечислових полів.

Приклад: Підрахувати кількість найменувань товару по кожній групі

```
SELECT P1.Найменування, Count(*) AS Кількість
FROM P1
GROUP BY P1.Група;
```

SUM – визначається сума всіх обраних значень даного поля, застосовується тільки для числових полів

Приклад: Визначити кількість кожного товару на складі

```
SELECT Найменування, sum(Кількість) as Всього
FROM склад
GROUP BY Найменування;
```

Всього – у даному випадку це ім'я поля, у якому відображатиметься сума значень поля Кількість для кожної групи у полі Найменування.

AVG - середньоарифметичне значення всіх обраних значень даного поля, застосовується тільки для числових полів.

Приклад: Визначити середню ціну кожного товару на складі

```
SELECT Найменування, avg(ціна) as ср_ціна
```

```
FROM склад
```

```
GROUP BY Найменування;
```

MIN – визначається найменше з всіх обраних значень даного поля, застосовується для числових і нечислових полів.

Приклад: Визначити найменування товару, який має мінімальну ціну

```
SELECT Найменування, min(ціна) as min_ціна
```

```
FROM склад;
```

MAX - найбільше з всіх обраних значень даного поля, застосовується для числових і нечислових полів

Крім функції COUNT (*) при обчисленні результату спочатку виключаються порожні рядки, після чого зазначена операція застосовується тільки до конкретних значень стовпця, що залишилися. Варіант COUNT (*) є особливим випадком функції COUNT () – у цьому випадку підраховуються всі рядки в результуючій таблиці, незалежно від того є чи порожні, дублюючі або інші значення.

Якщо до застосування узагальнюючої функції необхідно виключити дублюючі значення, впливає перед ім'ям стовпця у визначенні функції використовувати ключове слово **DISTING**. Це слово не може бути використане з функціями **MAX** і **MIN**. Однак його використання перед функціями **SUM** і **AVG** може привести до зміни результату. Крім того ключове поле **DISTING** може бути використане в запиті тільки один раз.

Узагальнюючі функції можуть бути використані тільки в списку пропозиції **SELECT** або **HAVING**.

Якщо список у **SELECT** містить узагальнюючі функції, і в текс-

ті запиту немає пропозиції **GROUP BY**, то жоден з елементів списку пропозиції **SELECT** не може містити посилань на стовпці, крім випадку, коли це посилання є аргументів узагальнюючої функції.

Приклад: Визначити кількість студентів, що успішно склали сесію

```
SELECT COUNT(*) as Кількість
FROM Сесія
WHERE Оцінка > 2;
```

Приклад: Визначити кількість студентів, що склали сесію і середній бал на сесії

```
SELECT COUNT(DISTING, Код_Студента) as Кількість,
AVG(Оцінка) as CP_бал
FROM Сесія
WHERE Оцінка IS NOT NULL;
```

Агрегатні функції не можна використовувати в пропозиції **WHERE**, тому що предикати оцінюються в термінах одиночного рядка, а агрегатні функції — у термінах груп рядків.

Пропозиція HAVING

У розділі **HAVING** задаються обмеження на виконання групування.

HAVING <Предикат-умова для групи>.

Пропозиція **HAVING** використовується разом із пропозицією **GROUP BY** для завдання обмежень на добір груп, що будуть поміщені в результуючу таблицю. Пропозиція **HAVING** використовується для фільтрації груп, на відміну від пропозиції **WHERE**, що використовується для фільтрації рядків.

У відповідності зі стандартом **ISO** усі імена стовпців пропозиції **HAVING** мають зазначатись у пропозиції **GROUP BY** або застосовуватися в узагальнюючій функції.

Приклад: визначити номери деталей, якщо сумарний розмір поставки перевищує 400.

```
SELECT DetNum, SUM(Det.Volume) AS SUM
From Det
```

```
GROUP BY DetNum
HAVING SUM(Det.Volume)>400;
```

Вкладені запити

Мова **SQL** дозволяє використовувати один запит у якості допоміжного для визначення результатів другого запиту.

Зовнішній оператор **SELECT** використовує результат внутрішнього запиту для визначення змісту остаточного результату всієї операції. Внутрішні запити можуть бути поміщені пропозиція **WHERE** і **HAVING** зовнішнього оператора **SELECT**, крім того внутрішні запити можуть використовуватися в операторах **INSERT**, **UPDATE**, **DELETE**

Існує три типи внутрішніх запитів:

- скалярний підзапит повертає єдине значення. Скалярний запит може використовуватися скрізь, де необхідно одержати єдине значення

- рядковий підзапит повертає значення декількох стовпців таблиці, але у виді єдиного рядка. Рядковий підзапит може використовуватися скрізь, де застосовується конструктор рядкових значень.

- табличний підзапит повертає значення одного або більш стовпців таблиці, розміщені більш ніж в одному рядку. Табличний підзапит може використовуватися скрізь, де допускається вказувати таблицю.

Вкладений запит являє собою інструмент створення тимчасової таблиці, зміст якої витягається й обробляється зовнішнім оператором. Вкладений запит може вказуватися безпосередньо після оператора порівняння (=, <, > і т.д.) у пропозиціях **WHERE** і **HAVING**. Текст підзапита має розміщуватись у дужках.

Приклад: Визначити список постачальників, статус яких менше максимального статусу в таблиці постачальників (порівняння з підзапитом):

```
SELECT *
FROM P
WHERE P.STATUS <
(SELECT MAX(P.STATUS)
FROM P);
```

Оскільки поле P.STATUS порівнюється з результатом підзапиту,

підзапит повинний бути сформульований так, щоб повертати єдине значення..

Використання вкладеного запиту з узагальнюючими функціями

Приклад Скласти список усіх співробітників, що одержують зарплату вище середньої по підприємству з указівкою, на скількох цих зарплат більше середньої.

```
SELECT
Кодработника, ФИО, Платня, (SELECT avg(зарплата) FROM
Штат;)
FROM Штат
WHERE Платня >
(SELECT avg(зарплата)
FROM Штат;);
```

У цьому випадку необхідно використовувати вкладений запит, тому що не можна використовувати узагальнюючі функції в пропозиції WHERE. Вкладений запит повертає єдине значення.

Використання предиката IN.

Використовується у випадку, коли вкладений запит повертає таблицю

Приклад: Одержати список постачальників, що поставляють деталь номер 2:

```
SELECT *
FROM P
WHERE P.PNUM IN
(SELECT DISTINCT PD.PNUM
FROM PD
WHERE PD.DNUM = 2);
```

У даному випадку вкладений вкладений запит може повертати таблицю, що містить кілька рядків.

Використання предиката EXIST.

Приклад: Одержати список постачальників, що поставляють деталь номер 2:

```
SELECT *
FROM P
WHERE EXIST
(SELECT *
FROM PD
```

```
WHERE
PD.PNUM = P.PNUM AND PD.DNUM = 2;);
```

Вкладений запит повертає таблицю значень.

Використання вкладеного запиту з перевіркою рівності

Приклад: Скласти список персоналу, що працює у філії, розташованій за адресою вул.Горького, 5

```
SELECT *
FROM Штат
WHERE Кодфіліала=
(SELECT Кодфіліала
FROM Філія
WHERE адреса ='вул.Горького,5');
```

В внутрішньому запиті визначається номер філії, яка розташована за зазначеною адресою. В зовнішньому запиті визначається список працівників даної філії.

Разом з внутрішнім запитом можна використовувати предикат EXISTS, що повертає істину, якщо висновок підзапиту не порожній.

Багато таблицні запити

У мові **SQL** операція з'єднання використовується для об'єднання інформації з двох таблиць шляхом утворення двох зв'язаних рядків, обраних з кожної таблиці.

Для виконання з'єднання досить у пропозиції **FROM** вказати імена декількох таблиць через кому, а в пропозиції **WHERE** указати стовпці, використовувані для з'єднання таблиць. При цьому якщо у схемі бази даних встановлений зв'язок між таблицями, які використовуються у запиті, то при з'єднанні здійснюється один з типів з'єднання таблиць, що передбачено у мові **SQL**. У **MS Access** за замовчуванням здійснюється внутрішнє з'єднання. Якщо у схемі бази даних зв'язок не встановлений, то здійснюється декартове множення таблиць за з'єднуваними рядками.

Найчастіше багато таблицні запити застосовуються для отримання інформації з таблиць, що мають зв'язок 1:M.

Синтаксис з'єднаних таблиць

При виконанні операції з'єднання дані з двох таблиць комбінуються з утворенням зв'язаних рядків, у яких значення порівнянних

Синтаксис з'єднаної таблиці має вигляд:

Таблиця А [тип з'єднання] JOIN Таблиця В

JOIN ... ON - З'єднання за допомогою предиката з'єднує рядки таблиць А та В за допомогою зазначеного предиката.

OUTER - Ключове слово OUTER (зовнішній) не є обов'язковими, воно не використовується ні в яких операціях з даними.

INNER - Тип з'єднання "внутрішнє". Внутрішній тип з'єднання використовується за замовчуванням, коли тип явно не заданий. У таблицях А та В з'єднуються тільки ті рядки, для яких знайдено збіг.

Приклад: Створити список студентів, що мешкають у гуртожитку, із указівкою всіх даних студентів

```
SELECT Студенти.*, Гуртожиток.Гуртожиток, Гуртожиток.Кімната
```

```
FROM Студенти INNER JOIN Гуртожиток ON Студенти.Код_студента = Гуртожиток.[Код студента];
```

LEFT (OUTER) - Тип з'єднання "ліве (зовнішнє)". Ліве з'єднання таблиць А і В містить у собі всі рядки з лівої таблиці А і ті рядки з правої таблиці В, для яких виявлений збіг. Для рядків з таблиці А, для яких не знайдено відповідності в таблиці В, у стовпці, що витягаються з таблиці В, заносяться значення NULL.

Приклад: Перелічити філії компанії й об'єкти, що здаються в оренду, що розташовані в тому самому місті, а також інші філії компанії

```
SELECT b.*, p.*
```

```
FROM філія b LEFT JOIN Оренда a on b.city=a.city
```

У результуючу таблицю попадають не тільки ті рядки, що знайшли собі пари в таблицях b і p, але і ті рядки з лівої таблиці, що з'єднуються, що не знайшли відповідності в другій таблиці. У цьому рядку всі поля правої таблиці заповнюються значеннями NULL.

RIGHT (OUTER) - Тип з'єднання "праве (зовнішнє)". Праве з'єднання таблиць А і В містить у собі всі рядки з правої таблиці В і ті рядки з лівої таблиці А, для яких виявлений збіг. Для рядків з таблиці В, для яких не знайдено відповідності в таблиці А, у стовпці, що витягаються з таблиці А заносяться значення NULL.

FULL (OUTER) - Тип з'єднання "повне (зовнішнє)". Це комбінація лівого і правого з'єднань. У повне з'єднання включаються всі ря-

дки з обох таблиць. Для співпадаючих рядків полючи заповнюються реальними значеннями, для незбіжних рядків полючи заповнюються відповідно до правил лівого і правого з'єднань.

UNION JOIN - З'єднання об'єднання є зворотним стосовно внутрішнього з'єднання. Воно включає тільки т рядки з таблиць А и В, для яких не знайдене збігів. У них використовуються значення NULL для стовпців, отриманих з іншої таблиці. Якщо взяти повне зовнішнє з'єднання і видалити з нього рядка, отримані в результаті внутрішнього з'єднання, то вийде з'єднання об'єднання.

2 ЛАБОРАТОРНЕ ЗАВДАННЯ

Всі створені запити зберегти за ім'ям відповідного завдання, наприклад, запит у завданні 2.1.1 зберегти за ім'ям 2_1_1.

Завдання 2. 1 Створення простого запиту на мові SQL

2.1.1. Визначити номери мобільних телефонів кожного викладача, у списку мають вказуватись й викладачі, у які не мають таких телефонів(або телефонів обраного оператора.

2.1.2. Створити список викладачів кафедри , які не мають телефону обраного оператора, список відсортувати за абеткою.

2.1.3. Створити список студентів своєї групи ,список має бути відсортований за абеткою.

Завдання 2. 2 Створення SQL запитів з використанням узагальнюючих функцій

2.2.1. Визначити кількість студентів 2000 року народження.

2.2.2. Визначити кількість викладачів, яки народились між 1980 та 1990 роками.

2.2.3. Визначити кількість студентів, яким нарахована стипендія у кожному місяці та її середній розмір.

2.2.4. Підрахувати кількість студентів, які мешкають у кожному гуртожитку.

2.2.5. Визначити кількість студентів, що мешкають у кожній кімнаті кожного гуртожитку.

Завдання 2. 3 Створення багато таблицного запиту

2.3.1. Створити список мешканців кожної кімнати у гуртожитку 1.

2.3.2. Визначити список студентів, які отримали хоча б одну двійку на іспитах.

2.3.3. Визначити середній бал з дисципліни Обчислювальна техніка та програмування .

2.3.4. Змінити запит за п. 2.3.3 таким чином, що б в результаті виконання запиту визначався середній бал з кожної дисципліни.

2.3.5. Визначити середній розмір стипендії у кожній групі по місяцях.

Завдання 2. 4 Використання вкладених запитів

2.4.1.Визначити список студентів, які мають оцінку з Обчислювальна техніка та програмування вище середнього балу з предмету.

2.4.2.Визначити список студентів, які не мешкають у гуртожитку

2.4.3.Визначити, які студенти, що мешкають у гуртожитку, склали іспити на 2

Завдання 2. 5 Створення запиту на оновлення

2.5.1. Змінити назву дисципліни Обчислювальна техніка та програмування на Програмування.

Додаткові завдання для самостійної розробки запитів

Варіант 1.

1. Знайти, на якій кафедрі не має жодного професора.
2. У якому гуртожитку проживають студенти з кожної групи?
3. Підрахувати кількість доцентів на усіх кафедрах.
4. Скільки доцентів приймало іспити протягом сесії?
5. Знайти викладача, якій поставив найбільшу кількість 2.
6. Знайти, в який день на сесії складали іспит більше двох груп.
7. Знайти, в який день на сесії всі студенти отримали найменший середній бал.
8. Знайти групу в якій стипендію будуть одержувати найбільша кількість студентів.
9. У якому гуртожитку не будуть отримувати стипендію найбільша кількість студентів?

Варіант 2.

1. Знайти, на якій кафедрі не має жодного доцента.
2. У якому гуртожитку проживають студенти не з усіх груп?
3. Підрахувати кількість професорів на усіх кафедрах.
4. Підрахувати кількість доцентів, які приймали іспити на сесії.
5. Знайти викладача, якій поставив найбільшу кількість 5.
6. Знайти в який день на сесії складала іспит тільки одна група.
7. Знайти, в який день на сесії складали іспит найбільша кількість студентів.
8. Знайти групу в якій стипендію будуть одержувати найменша кількість студентів.

9. У якому гуртожитку не будуть отримувати стипендію найменша більша кількість студентів?

Варіант 3.

1. Знайти, на якій кафедрі не має жодного старшого викладача.
2. У якому гуртожитку проживає найбільша кількість студентів?
3. Підрахувати кількість старших викладачів на усіх кафедрах.
4. Підрахувати кількість професорів, які приймали іспити протягом сесії.
5. Знайти викладача, якій на сесії поставив 4 більше за всіх.
6. Знайти, в який день на сесії складала іспит найменша кількість студентів.
7. Знайти, в який групі студенти отримали найбільшу кількість
- 5.
8. Знайти групу в якій середня стипендія за вересень буде найбільша.
9. У якому гуртожитку студенти будуть отримувати найменшу середню стипендію за жовтень?

Варіант 4.

1. Знайти, на якій кафедрі є викладачі професори.
2. У якому гуртожитку проживає найменша кількість студентів?
3. Підрахувати кількість викладачів пенсійного віку на усіх кафедрах.
4. Підрахувати кількість старших викладачів, які приймали іспити протягом сесії.
5. Знайти викладача, якій на сесії поставив найбільше 3.
6. Знайти, в який день на сесії всі студенти отримали найменшу кількість 5.
7. Знайти, в який групі студенти отримали найбільший середній бал.
8. Знайти групу в якій середня стипендія за жовтень буде найменша.
9. У якому гуртожитку студенти будуть отримувати найбільшу середню меншу стипендію за жовтень?

Вариант 5.

1. Знайти, на яких кафедрах є викладачі професори.
2. У якому гуртожитку проживає найменша кількість студентів, які отримали на сесії хоча б одну 2?
3. Підрахувати на якій кафедрі найменша кількість викладачів.
4. Підрахувати кількість іспитів, які склали студенти протягом сесії.
5. Знайти викладача, якій на сесії не поставив жодної 2.
6. Знайти, в який день на сесії студенти отримали найбільшу кількість 2.
7. Знайти, в який групі студенти отримали найбільший середній бал.
8. Знайти групу в якій найбільша кількість студентів, що отримують стипендію протягом всього семестру.
9. У якому гуртожитку проживають студенти що не будуть отримувати стипендію протягом семестру?

Вариант 6.

1. Знайти, скільки професорів на всіх кафедрах.
2. У якому гуртожитку проживає найменша кількість студентів, які отримали на сесії тільки 4 та 5?
3. Підрахувати на якій кафедрі найбільша кількість викладачів.
4. Підрахувати кількість іспитів, які прийняли викладачі протягом сесії.
5. Знайти викладача, якій на сесії не поставив жодної 5.
6. Знайти, в який день на сесії всі студенти отримали найбільший середній бал.
7. Знайти, в який групі найбільшу кількість студентів, які склали іспити тільки на 4 та 5.
8. Знайти групу в якій найменша кількість студентів, що отримують стипендію протягом всього семестру.
9. Підрахувати кількість студентів у кожному гуртожитку, що отримувати стипендію більшу за середню у вересні.

3 ЗМІСТ ЗВІТУ

- 3.1.Тема та мета лабораторної роботи.
- 3.2.Тексти всіх SQL запитів та результати їх виконання.
- 3.3.Відповіді на контрольні питання.

4 КОНТРОЛЬНІ ПИТАННЯ

- 4.1.Основні пропозиції оператору SELECT ?
- 4.2.Як у мові SQL визначаються умови добору рядків?
- 4.3.Як у мові SQL визначається сортування даних?
- 4.4.Як створити обчислювальне поле у SQL запиті?
- 4.5.Призначення пропозиції Having у операторі SELECT
- 4.6.Які умови можна використовувати у пропозиції WHERE?
- 4.7.Як з'єднати дані двох таблиць, які мають однаковий стовпчик?
- 4.8.Що таке вкладений запит? У яких випадках використовується вкладений запит?

Лабораторна робота № 2 **Створення кнопкової форми**

Мета: вивчення процесу створення кнопкових форм для роботи з базою даних, за допомогою Диспетчера кнопкових форм.

1 ЗАГАЛЬНІ ВІДОМОСТІ

Кнопкова форма

Кнопкова форма є не що інше, як Меню для роботи в базі даних. У меню може бути декілька вкладених підменю.

При створенні бази даних за допомогою майстра автоматично створюється кнопкова форма, що допомагає переміщатися по базі даних. На кнопкову панель поміщаються кнопки, при натисканні яких відкриваються форми або звіти (або відкриваються інші кнопкові форми, за допомогою яких відкриваються інші форми або звіти), здійснюється вихід з Microsoft Access або змінюється сама кнопкова форма. Диспетчер кнопкових форм дозволяє створювати кнопкові форми, подібні тим, що створює майстер баз даних.

Створення кнопкової форми за допомогою диспетчера кнопкових форм

Для створення кнопкової форми за допомогою диспетчера кнопкових форм треба виконати наступну послідовність дій:

1. Вибрати у меню Сервіс → команду Настройки → Диспетчер Кнопочных форм.
2. Якщо виводиться запрошення, підтвердити створення кнопкової форми, натиснути кнопку Да.
3. У діалоговому вікні Диспетчер кнопочных форм натиснути кнопку Изменить.
4. У діалоговому вікні Изменение сторінки кнопкової форми ввести ім'я в поле Название кнопочной формы й натиснути кнопку Создать.

5. У діалоговому вікні Изменение страницы кнопочной формы ввести текст підпису для першої кнопки й вибрати команду в поле зі списком Команда. Наприклад, увести текст Перегляд замовлень, а потім вибрати у поле Команда команду Открытие формы в режиме редактирования.

6. Для більшості обраних команд під полем Команда відкривається нове поле зі списком. При необхідності, вибрати потрібний елемент у цьому полі. Наприклад, якщо обрано одна з команд роботи з формами, то в поле зі списком Форма варто вибрати ім'я відкриваємої форми.

7. Повторити кроки з 4 по 6 для створення всіх потрібних кнопок кнопкової форми. Для зміни або видалення якої-небудь зі створених кнопок, вибрати ім'я цієї кнопки в списку Элементы даної кнопкової форми й натиснути кнопку Изменить або Удалить. При необхідності змінити порядок кнопок у списку вибрати елемент і натисніть кнопку Вверх або Вниз.

8. Після закінчення створення кнопкової форми, натиснути кнопку Закрьть.

Диспетчер кнопкових форм дозволяє створювати кнопкову форму, кнопки якої відкривають інші кнопкові форми. За допомогою описаних вище дій можуть бути створені кілька кнопкових форм. Для того щоб зробити кнопкову форму стартовою кнопковою формою, що відкривається за замовчуванням при запуску бази даних, вибрати ім'я кнопкової форми в діалоговому вікні Диспетчер кнопочных форм і натиснути кнопку По умолчанию. Для того, щоб відкрити із кнопкової форми іншу кнопкову форму, вибрати на кроці 5 команду Переход к кнопочной форме й указати ім'я кнопкової форми, на яку потрібно перейти.

При створенні кнопкової форми за допомогою диспетчера кнопкових форм Microsoft Access створює таблицю Элементы кнопочной формы, яка містить опис кнопок, що виводяться у формі, і виконуваних ними дій. Спроба змінити кнопкову форму в режимі конструктора форми, може привести до того, що додаток перестане працювати. Якщо потрібно внести в кнопкову форму істотні зміни, рекомендується створити кнопкову форму заново, а потім указати нову форму як стартовою формою.

Оформлення Меню для роботи з базою

Меню роботи з базою можна зовні змінити, доповнивши його малюнками, значками, змінити розмір кнопок, змінити шрифт написів пунктів меню і т.і. Крім того створивши Меню роботи з базою бажано від сторонніх очей сховати вікно бази даних, щоб користувач розробленого програмного продукту не зміг побачити й змінити таблиці, запити, макроси.

Змінити шрифт написів пунктів меню:

- Відкрити кнопку форму в режимі конструктора
 - Виділити мишою те місце, де може бути напис пункту меню “Формы”
 - Вибрати пункт меню “Вид\свойства”, (або просто клацнути правою кнопкою миші)
 - Відкрийте вкладку “Макет” і можете на свій розсуд змінити: розмір шрифту; кольори тексту; кольори границі; насиченість.
- Аналогічно можна змінити оформлення для всіх пунктів меню.

Вставка малюнок у кнопку форму на порожнє місце:

- Відкрити кнопку форму в режимі конструктора
 - Клацніть правою кнопкою по порожньому місцю праворуч чи ліворуч від меню
 - Вибрати пункт “Свойства”
 - Вибрати вкладку “Макет/ Рисунок”, Натиснути кнопку ...
 - Вибрати малюнок у папці C:\Windows
- Оформлення можна продовжувати далі, змінюючи колірні гами місця, що залишилося, на панелі Меню.

Сховати вікно бази даних:

- Вибрати пункт меню “Сервис / Параметры запуска”
- Зняти прапорець “Окно базы данных”
- Закрити базу даних, і відкрити її знову, перед Вами відразу з'явиться Меню.

Стискування бази даних при виході з додатка

Для підвищення продуктивності додатка, якій створено, при виході з програми необхідно стиснути та відновити базу даних. Для цього треба виконати наступні дії:

- вибрати в командному меню Сервис → Служебные программы→ Сжать и восстановить базу данных.
- Для постійного виконання стискування бази при виході з додатка треба встановити прапорець Сервис→ Параметры→ Общие → Сжимать при закрытии

2 ЛАБОРАТОРНЕ ЗАВДАННЯ

2.1 Приклад розробки кнопкової форми

Створити меню, яке має 3-х рівневу структуру.

2.1.1. Створення меню 1-го рівня:

- Закрити всі таблиці, форми, звіти, запити
- Вибрати пункт меню “Сервис/ Служебные программы/ Диспетчер кнопочных форм”, в відповідь на питання “Создать кнопочную форму” натиснути кнопку Да. Відкривається вікно диспетчера кнопочних форм
 - Натиснути кнопку “Изменить”
 - Ввести назву кнопкової форми “Деканат”
 - Натиснути кнопку Закреть
 - Натиснути кнопку Создать
 - У вікні “Создание” ввести текст “Форми”→ ОК
 - Натиснути кнопку Создать
 - У вікні “Создание” ввести текст “Запити”→ ОК
 - Натиснути кнопку Создать
 - Аналогічно створити наступні сторінки кнопкової форми: “Звіти”, ”Вихід”

2.1.2. Створення меню 2-го рівня:

Для послідовного розкривання кожен пункт меню 1-го рівня виконати наступні дії:

- Помітити курсором головну кнопкову форму ”Деканат” и натиснути кнопку Изменить
- Створити елемент кнопкової форми
- Текст “Форми”

- Команда “Переход к кнопочной форме”
- Кнопкова форма - вибрати з списку ”Форми”→ ОК.
- Створити елемент кнопкової форми
- Текст “Запити”
- Команда “Переход к кнопочной форме”
- Кнопкова форма - вибрати з списку ”Запити”→ ОК
- Створити елемент кнопкової форми
- Текст ”Звіти”
- Команда “Переход к кнопочной форме”
- Кнопкова форма - вибрати з списку ”Звіти”→ ОК
- Створити елемент кнопкової форми
- Текст “Вихід”
- Команда ”Вихід з Access”
- Кнопкова форма - вибрати з списку “Выход”, ОК

При закритті вікна “Изменение страницы кнопочной формы” знов відкривається вікно “Диспетчер кнопочных форм”

2.1.3. Створити меню 3-го рівня:

Установити курсор на сторінку ”Форми” Натиснути кнопку Изменить

У вікні “Изменение страниц кнопочной формы” виконати наступні дії:

- Створити елемент кнопкової форми
- Текст “Викладачі”
- Команда “Открытие формы в режиме редактирования”
- Кнопкова форма - вибрати з списку “Викладачі”→ ОК
- Створити елемент кнопкової форми
- Текст “Підсумки здачі іспиту в групі”
- Команда “Открытие формы в режиме редактирования”
- Кнопкова форма - вибрати з списку “Підсумки здачі іспиту в групі”, ОК
- Створити елемент кнопкової форми
- Текст “Список студентів по групах”
- Команда “Открытие формы в режиме редактирования”
- Кнопкова форма - вибрати з списку “Список студентів по групах”, ОК

- Аналогічно створити елемент кнопкової форми для перегляду форм: "Підсумки здачі інформатики", "Підсумки сесії"

2.2 Самостійна розробка кнопкової форми

2.2.1 Створити меню 3-го рівня виклику запитів, які є у вашій базі даних (створені у л.р. „Створення запитів”), виконавши дії аналогічні діям при створенні меню для форм.

2.2.2 Створити меню 3-го рівня для перегляду звітів (створені у лабораторній роботі „Створення звітів”), виконавши дії аналогічні діям при створенні меню для форм і запитів.

2.2.3 У Вашому Меню є одна кнопка виходу, настроєна на вихід з бази даних. Створіть на кожному рівні кнопки керування для повернення на більш високий рівень (вихід у меню “Деканат”).

2.2.4 Змінити оформлення Меню для роботи з базою, доповнивши його:

- малюнками та значками
- змінити розмір кнопок
- змінити шрифт написів пунктів меню

2.2.5 Зробити розроблену кнопку форму стартовою.

Встановити режим стискування бази даних при виході з додатка.

2.2.6 Сховати вікно бази даних.

3 ЗМІСТ ЗВІТУ

3.1. Тема та мета лабораторної роботи.

3.2. Послідовність виконання завдань лабораторної роботи для самостійного виконання.

3.3. Відповіді на контрольні питання.

4 КОНТРОЛЬНІ ПИТАННЯ

4.1. Що таке кнопкова форма?

4.2. Як створити кнопку форму?

4.3. Як зробити кнопку форму стартовою?

- 4.4. Як змінити шрифт написів пунктів меню?
- 4.5. Як вставити малюнок у кнопку форму?
- 4.6. Як сховати вікно бази даних?

РЕКОМЕНДОВАНА ЛІТЕРАТУРА

- 5.1. Элен Феддема, Access 2016, Питер, Москва-СПб, 2017, 940с.
- 5.2. С Робинсон, Microsoft Access, Учебный курс, Питер, СПб-Москва-Харьков-Минск, 2010, 512с.
- 5.3. Куртер, Office 2010. Учебный курс, СПб., Питер Пресс, 2010, 680 с.
- 5.4. Т.Коннолли, К Бегг, А. Страчан, Базы данных, проектирование, реализация и сопровождение. Теория и практика, М-СПб_Киев, 2010, 1112с.