Міністерство освіти і науки України
Національний університет «Запорізька політехніка»

БАЗИ ДАНИХ

методичні вказівки до лабораторних робіт. Частина 1

для студентів спеціальності
121 «Інженерія програмного забезпечення»
усіх форм навчання

2021

Бази даних. Методичні вказівки до лабораторних робіт. Частина 1 для студентів спеціальності 121 «Інженерія програмного забезпечення» усіх форм навчання / Уклад. С.К. Корнієнко – Запоріжжя: НУ «Запорізька політехніка», 2021. – 46 с.

Укладач: С.К. Корнієнко, доцент, к.т.н.

Рецензент: Г.В. Шило, доцент, д.т.н.

Відповідальний
за випуск: С.О. Субботін, професор, д.т.н.

Затверджено
на засіданні кафедри
програмних засобів

Протокол № 1 від 18.08.2020 р.

ЗМІСТ

1 ЛАБОРАТОРНА РОБОТА № 1 «СТВОРЕННЯ СХЕМИ БАЗИ ДАНИХ»

1.1 Мета роботи

Метою роботи є ознайомлення з пакетом MS Access, придбання початкових навичок роботи з ним і створення схеми бази даних.

1.2 Завдання до лабораторної роботи

1.2.1 Ознайомитися зі змістом пункту 1.3 даних методичних указівок.

1.2. Відповідно до індивідуального завдання розробити схему бази даних за допомогою Конструктора таблиць та мови SQL.

1.2.3 Ввести дані.

1.3 Основні теоретичні відомості

1.3.1 Специфікація структури таблиці

При проектуванні бази даних треба визначити, яка саме інформація повинна входити до бази даних і чи повинна вся збережена в базі даних інформація розташовуватися в одній таблиці або її краще розділити на декілька таблиць.

Створення таблиці виконується у наступним чином

– запустіть MS Access;

– задайте ім'я файлу бази даних і натисніть кнопку **Створити**;

– у вікні бази даних, що з'явилося, виберіть закладку **Таблиці** та натисніть кнопку **Створити**. На екрані з'явиться вікно **Створення таблиці**, у правій частині якого знаходиться список можливих засобів створення таблиці;

– виберіть засіб **Конструктор** і, натиснувши кнопку **ОК**, приступайте до проектування нової таблиці за допомогою конструктора таблиць

Специфікація таблиці містить опис полів. Для кожного поля необхідно зазначити його ім'я та тип даних (типи даних, які використовуються в Access, наведені в таблиці 1.1). Можна також прокоментувати призначення поля в колонці **Опис.**

Таблиця 1.1 - Типи даних Access

Тип даних	Опис
Короткий текст (Текстовий)	Текст, максимальна довжина якого дорівнює 255 символам або визначається значенням властивості «Розмір поля»
Довгий текст (Поле МЕМО)	Текст, максимальна довжина якого дорівнює 1 Гбайт. Слід мати на увазі, що у формах та звітах відображаються лише перші 64 Кбайт.
Числовий	Числові дані, які використовуються для проведення розрахунків (1, 2, 4 або 8 байт): байт – 1 байт; ціле – 2 байти, довге ціле – 4 байти; числових значень із переміщуваною комою – 4 або 8 байтів.
Дата/час	Дати та час (8 байт)
Грошовий	Числа, що зберігаються з точністю до 15 знаків у цілій та до 4 знаків у дробовій частині (8 байт)
Лічильник	Число, що автоматично збільшується при додаванні в таблицю кожного нового запису. Значення цих полів не можуть бути змінені (4 байта)
Логічний	Логічні значення, а також поля, що можуть містити одне з двох припустимих значень.
Поле об'єкта OLE	Зображення, графічні об’єкти або інші об’єкти ActiveX з інших програм на платформі Windows. (до 2 Гбайт).
Обчислюваний	Вираз, який використовує дані з одного або кількох полів. На основі виразу можна призначити різні типи даних результату. Зверніть увагу, що файли у форматі MDB не підтримують тип даних «Обчислюваний».
Гіперпосилання	Адреса посилання на документ або файл в Інтернеті, інтрамережі, локальній мережі або на локальному комп’ютері.
Вкладення	Зображення, документи, електронні таблиці або діаграми та інші файли.. Зверніть увагу, що файли у форматі MDB не підтримують тип даних «Вкладення».
Майстер підстановок	Відображає дані, що підставляються з іншої таблиці або списку, що задається користувачем. При цьому запускається майстер підстановок, за допомогою якого визначається тип даних і організується зв'язок з іншою таблицею.

Для нормальної роботи з БД потрібна однозначна ідентифікація кожного запису даних, для чого одне з полів призначається у якості ідентифікатора (або **первинного ключа**).

За допомогою піктограми ключа на стандартній панелі інструментів можна оголосити потрібне поле ключовим.

1.3.2 Властивості полів

Вкладка «Загальні»

Після визначення імені та розміру поля вказують його властивості на вкладці «**Загальні»** в лівому нижньому куті екрану. Опис властивостей полів наведений у таблиці 1.2.

Таблиця 1.2 - Властивості полів Access

Властивість	Пояснення
Розмір поля	Текстовий: обмежує розмір поля заданим числом символів; по замовчанню приймається значення 50. Числовий: дозволяє задавати тип числа.
Формат поля	Змінює зовнішній вигляд даних після вводу значень.
Маска вводу	Використовується для організації вводу даних у жорстко визначеному форматі (номера телефонів, поштові індекси, тощо).
Число десяткових знаків	Задає кількість десяткових знаків (тільки в даних числового й грошового типів).
Підпис	Необов'язковий ярлик для полів форм і звітів (що замінює ім'я поля).
Значення по замовчанню	Значення, що автоматично з'являється в полі перед введенням даних.
Умова на значення	Забезпечує перевірку припустимості даних на підставі правил, які створені за допомогою виразів або макросів.
Повідомлення про помилку	Повідомляє про неприпустимість уведеного значення (задається користувачем).
Обов'язкове поле	Визначає, чи обов'язково вводити в цьому полі.
Індексоване поле	Прискорює доступ до даного та при необхідності обмежує дані, що вводяться, тільки унікальними значеннями.

Формати дозволяють указувати спосіб відображення на екрані тексту, чисел, значень дат і часу. Деякі типи даних мають стандартний формат, інші – тільки визначений користувачем формат, а треті – обидва формати. **Формати впливають тільки на відображення даних, а не на спосіб їхнього вводу або зберігання в таблиці**.

Для завдання форматів користувача у полях даних типу **Текстовий** і **Memo** використовуються чотири символи (таблиця 1.3).

Таблиця 1.3 - Символи завдання форматів користувача

Символ	Опис
@	обов'язковий текстовий символ або прогалина
&	необов'язковий текстовий символ
>	перетворить усі символи в прописні
<	перетворить усі символи в рядкові

Символи **@** і **&** впливають на окремі символи, а **>** та **<** – на усі введені символи. Якщо, наприклад, необхідно, щоб введене ім'я відображалося прописними буквами, то треба ввести значення **>** для властивості **Формат поля**. При введенні, наприклад, номера телефону можна задати такий формат **(**@@@**)**@@@-@@-@@. Якщо потім увести 612344586, то дані будуть відображені у вигляді (067)354-45-86.

Маска призначена для полегшення вводу даних у поле (табл. 1.4).

Таблиця 1.4 – Символи, які використовуються в масці вводу

Символ	Опис
1	**2**
0	Цифра (0-9, обов'язковий символ; знаки плюс (+) і мінус (–) не дозволені)
9	Цифра або прогалина (0-9, необов'язковий символ; знаки плюс (+) і мінус (–) не дозволені)
#	Цифра, знаки плюс (+), мінус (–) або прогалина (необов'язковий символ; незаповнені позиції виводяться як прогалини в режимі редагування, але вилучаються при зберіганні даних)

Продовження таблиці 1.4

1	2
L	Буква (обов'язковий символ)
?	Буква (необов'язковий символ)
A	Буква або цифра (обов'язковий символ)
a	Буква або цифра (необов'язковий символ)
&	Будь-який символ або прогалина (обов'язковий символ)
C	Будь-який символ або прогалина (необов'язковий символ)
<	Всі введені після нього символи перетворюються в рядкові
>	Всі введені після нього символи перетворюються в прописні
!	Вказує, що маска вводу заповнюється справа наліво. Варто використовувати, якщо в лівій частині маски знаходяться позиції, заповнювати котрі необов'язково. Символ можна розміщувати в будь-якій позиції маски вводу.
\	Наступний за ним символ відображається в такому ж вигляді, у якому він був уведений (наприклад, \{ призведе до появи символу {).
. , : ; - /	Десятковий роздільник, роздільники груп розрядів, часу або дати. (Використані символи роздільників визначаються у вікні **Мова та стандарти** панелі керування Windows).

Наприклад, для поля **Телефон** можна задати таку маску вводу, що дозволяла б уводити тільки цифри й автоматично добавляла проміжні символи: (999)900-00-00. Access автоматично додасть символ "\" для кожного проміжного символу: \(999\)900\-00\-00.

Увага

Якщо визначити для даних маску вводу та властивість **Формат поля**, то при відображенні даних у Access властивість **Формат поля** буде мати перевагу. Це означає, що навіть після зберігання маски вводу разом із даними, вона буде ігноруватися при форматуванні даних.

Вкладка «Підстановка»

У цій вкладці знаходиться тільки властивість **Тип елементу керування**, для якого є три варіанти вибору: {Поле, Список, Поле зі списком).

При завданні **Списку** або **Поля зі списком** необхідно зазначити джерело, в якому зберігаються варіанти даних, які вводяться (Таблиця/ запит, ім'я стовпчика або стовпчиків). При введенні даних користувач може вибирати значення з готового списку, що полегшує введення інформації, а також підвищує її достовірність.

> **Примітка**
> **Поле зі списком** відрізняється від **Списку** тим, що дозволяє вводити значення, відсутні в списку.

Завдання типу підстановки здійснюється або вручну на відповідній вкладці, або за допомогою **Майстра підстановки**, що викликається в стовпчику **Тип даних**.

> **Примітка**
> Припускається також створення в таблиці поля підстановки із посиланням на інше поле в тієї ж таблиці.

1.3.3 Встановлення зв'язків, ключі

Для встановлення зв'язків між таблицями необхідно встановити зв'язок між тими полями, у яких міститься загальна (за сенсом) інформація. Ці поля можуть мати різні імена, але тип даних і довжина полів, а також (що особливо важливо) інформація в обох полях відповідних записів повинні бути однаковими в обох таблицях. Як правило, зв'язок установлюється з'єднанням **ключових полів** таблиць: **первинного ключа** однієї таблиці та **зовнішнього ключа** – в інший.

Кожна таблиця повинна містити **первинний ключ** – одне або декілька полів, уміст яких унікальний для кожного запису.

При створенні таблиць Access завжди пропонує задати первинний ключ із типом даних **Лічильник**, значення якого буде автоматично збільшуватися на одиницю при додаванні нового запису.

> **Примітка**
> Поля первинного ключа повинні бути як можна коротшими, оскільки їхні розміри впливають на швидкість виконання операцій у базах даних.

Зовнішні ключі. Поняття посилальної цілісності

Коли поле однієї таблиці посилається на інше поле (в іншій таблиці), воно зветься **зовнішнім ключем (foreign key)**. Поле, на який він посилається, зветься його **батьківським ключем (parent key)**. Імена зовнішнього та батьківського ключів можуть не співпадати, але тип і розмір даних повинні бути однаковими.

> **Примітка**
> Якщо ключ батьківської таблиці має тип Лічильник, то відповідний зовнішній ключ повинний бути довгим цілим.

Таким чином, **основна ідея посилальної цілісності** полягає в тому, що при введенні даного до зовнішнього ключа батьківський ключ перевіряється на наявність в ньому такого значення. При відсутності команда відхиляється.

1.3.4 Конструктор зв'язків

Конструктор зв'язків викликається командою **Робота з базами даних⇨Схема даних** або за допомогою кнопки **Схема даних**  панелі інструментів .

На початку роботи вікно **Схема даних** не містить таблиць.

Додавати таблиці до вікна можна у такий спосіб:

– використавши діалогове вікно **Додавання таблиці**; що відчиняється автоматично, якщо вікно **Схема даних** для бази даних відчиняється вперше;

– у вікні **Схема даних** викликати контекстне меню та вибрати команду **Додати таблицю.**

Встановлення зв'язків між таблицями

Перш за все, слід мати на увазі, що таблиці зв’язуються лише за полем, яке є загальним (за вмістом , а не за назвою) для обох таблиць. Для встановлення зв'язків між таблицями потрібно вибрати загальне поле в одній таблиці та перетягнути його мишкою в загальне поле тієї таблиці, яку необхідно зв'язати з першою. При цьому викликається діалогове вікно **Зв'язки**, показане на рисунку 1.1.

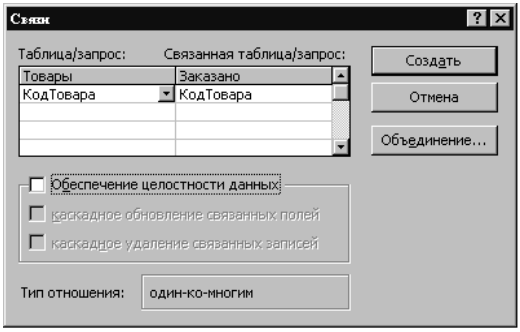


Рисунок 1.1 - Діалогове вікно Зв'язки

У вікні **Зв'язки** міститься інформація про поля зв'язку головної та підпорядкованої таблиць, параметрах забезпечення цілісності даних і тип зв'язку, який встановлюється.

У верхній частині вікна виводяться найменування таблиць. Таблиця, що знаходиться зліва, рахується первинною в цьому відношенні. Нижче розміщується поля зв'язку обох таблиць. Після встановлення зв'язку можна включити підтримку цілісності даних. **Цілісність даних** – це набір правил, які охороняють пов'язані таблиці від внесення некоректних значень.

Каскадне оновлення пов'язаних полів

Якщо включена опція **Забезпечення цілісності даних** у вікні **Зв'язки**, Access дозволяє включити опцію **Каскадне оновлення пов'язаних полів**. При цьому зміна значення ключа в батьківській таблиці автоматично викликає оновлення відповідних полів в підлеглих таблицях.

> **Увага**
> Якщо опція **Каскадне оновлення пов'язаних полів** не включена, змінити значення ключового поля первинної таблиці не можна.

Каскадне вилучання пов'язаних записів

Ця опція дає можливість після вилучання запису у первинній таблиці автоматично вилучати відповідні записи в підлеглих таблицях.

> **Увага**
> Будьте обережні з опцією вилучання пов'язаних записів! Access не попереджує про каскадне вилучання

Для вилучання існуючого зв'язку потрібно зайти у вікно **Схема даних**, виділити потрібний зв'язок і вилучити його за допомогою або кнопки на панелі інструментів, або команди **Вилучити** в контекстному меню.

1.3.5 Створення таблиць за допомогою мови SQL

Для створення таблиці використовується SQL-запит ***Create table***. Наприклад, треба створити таблиці **Покупці** та **Покупки**, між якими існує зв'язок 1:N.

Таблиця **Покупці** створюється за допомогою запиту:

```
create table Покупці
        (код  integer not null primary key,
        прізвище  char(15),
        адреса varchar(35) not null,
        phone char(8)  null);
```

Примітка

SQL являється мовою з *нефіксованими формами (free-form language),* тобто він не обмежує кількість слів в одному рядку та місця розривів рядків.

В інструкціях і запитах SQL пропуски ігноруються, дозволяючи форматувати код SQL з метою покращення читабельності.

Обмеження ***primery key*** означає первинний ключ.

Статус стовпчика ***NOT NULL*** означає обов'язкове заповнення відповідного стовпчика. Статус ***NULL*** – необов'язковість.

Типи даних ***char*** і ***varchar*** відрізняються тим, що для даних типу ***char*** виділяється дисковий простір фіксованої довжини незалежно від реальної потреби, а для даних типу ***varchar*** дисковий простір виділяється по мірі необхідності в рамках зазначеного розміру, що дозволяє заощаджувати дисковий простір.

Таблиця **Покупки** є підлеглою. Тому до її складу слід додати атрибут **покупець,** який буде виконувати роль зовнішнього ключа (***foreign key***):

```
create table Покупки
        (ном int not null primery key,
        покупець int not null,
        дата date not null,
        сума money not null,
        foreign key(покупець) references Покупці(код);
```

Для входу до режиму редагування SQL-запиту треба викликати **Конструктор запитів** у вкладці **Створення,** в якому слід зачинити вікно **Додавання таблиці**. Після цього у верхньому лівому куті екрану з'явиться кнопка **SQL**, натискання на яку викличе режим створення та редагування SQL-запиту.

Текст кожного запиту зберігається в окремому файлі.

Примітка

Запит на створення таблиці можна запускати один раз. При повторному запуску буде видана помилка, оскільки таблиця вже існує.

Контрольні питання

1.1 Які типи даних використовуються в Access?

1.2 Як задається режим автоматичної нумерації записів?

1.3 Як можна задати значення по замовчанню в числові поля?

1.4 Що таке формат даних, як він задається?

1.5 Для чого використовується маска вводу та як вона задається?

1.6 Як задається ключове поле?

1.7 Які властивості поля підтримує Access?

1.8 Як можна задати список значень для підстановки в поле?

1.9 Що таке первинні та зовнішні ключі?

1.10 Як встановлюються та вилучаються зв'язки між таблицями?

1.11 Поняття цілісності даних; як воно підтримується в Access?

1.12 Як встановлюється Каскадне вилучання пов'язаних полів?

1.13 Як встановити Каскадне оновлення пов'язаних полів?

2 ЛАБОРАТОРНА РОБОТА № 2 «ЗАПИТИ НА ВИБІРКУ»

2.1 Мета роботи

Метою роботи є надбання навичок створенням запитів на вибірку.

2.2 Завдання до лабораторної роботи

Для бази даних, створеної в попередній лабораторній роботі, за узгодженням із викладачем розробити різноманітні запити, використовуючи можливі варіанти їхнього створення.

2.3 Основні теоретичні відомості

За допомогою запитів користувач може вибрати з таблиці необхідні дані й відобразити їх на екрані. Результатом опрацювання запиту є таблиця, яка має назву **динамічного, тимчасового набору даних.**

При зберіганні запиту зберігається тільки структура запиту, тобто лише перелік таблиць, список полів, порядок сортування, обмеження на записи, тип запиту тощо. Динамічний набір даних при цьому не зберігається. При кожному виконанні запиту він будується знову на базі «свіжих» табличних даних.

Access підтримує такі основні типи запитів:

– вибірка (select);
– вилучання (delete);
– додавання (insert);
– оновлення (update).

В Access основними засобами побудови запитів є використання **Конструктора запитів**, вбудованої мови SQL або їхньої комбінації, тобто створення запиту за допомогою Конструктора з наступним корегуванням на мові SQL.

Запит на вибірку є найпоширенішим типом запиту. При виконанні запиту дані вибираються з однієї або декількох таблиць і результати відображає в режимі таблиці.

2.3.1 Використання Конструктору запитів

Для проектування запиту потрібно в меню Access вибрати ланцюжок **Створення → Інші → Конструктор запитів.**

На екрані з'явиться порожнє вікно створення запиту, а також діалогове багатосторінкове вікно **Додавання таблиці**, за допомогою якого можна вибрати таблиці або інші запити, що будуть використовуватися при створенні даного запиту.

Для вибору таблиць або запитів потрібно виділити відповідний елемент і натиснути кнопку **Додати**. По закінченні вибору треба натиснути кнопку **Закрити**.

Вікно Конструктора запитів складається з двох частин (рис. 2.1):

Область таблиць запиту – це місце, де розміщаються таблиці або запити й встановлюються зв'язки між ними.

Бланк запиту призначений для визначення полів і умов, що будуть використані для витягу динамічного набору даних. У кожному стовпчику бланка запиту міститься інформація про одне поле з таблиці або запиту у верхній частині екрана.

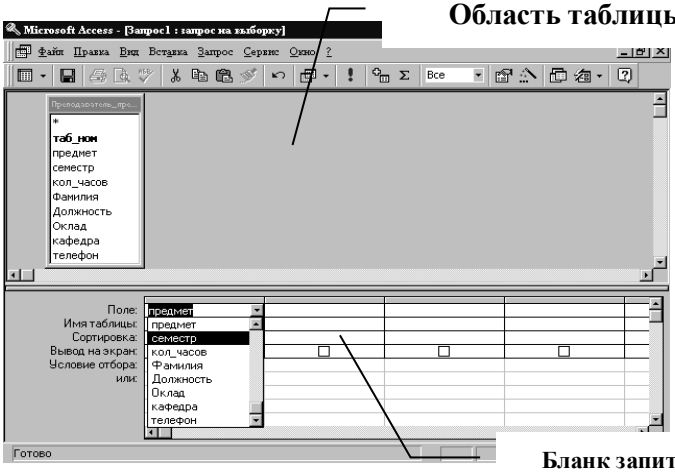


Рисунок 2.1 - Вікно конструктора запитів

Бланк запиту містить шість рядків:

1. **Поле** – ім'я поля;
2. **Ім'я таблиці –** ім'я таблиці;
3. **Сортування –** спосіб сортування;
4. **Виведення на екран –** визначає, чи буде присутнім поле в динамічному наборі даних;
5. **Умова відбору** – містить першу умову, що обмежує набір записів;
6. **Або** – інші умови обмеження набору записів.

Вставка та вилучання полів

Ім'я поля вводиться до бланка запиту або перетягуванням його з відповідної таблиці, або зі списку, що розкривається, у потрібному стовпчику бланка запиту. Щоб додати до бланка запиту всі поля таблиці, потрібно виділити поле (*) у таблиці, а потім перетягнути піктограму **Поле** до першої чарунки бланку запиту.

Для вилучання стовпчиків із бланка запиту можна скористатися командами **Правка⇨Вилучити стовпчики** або **Правка⇨Очистити бланк**, а також шляхом виділення стовпчика та натискання клавіші <**Del**>.

Зміна порядку сортування

Сортування даних за окремими полями може виконуватися як по зростанню, так і по зменшенню. Сортування за декількома полями залежить від порядку розміщення полів у бланку запиту (зліва на право).

> **Примітка**
> Поля типу **Memo** та **OLE** не сортуються

Відбір записів

Умови відбору записів – це набір визначених у Access або заданих користувачем правил. Умови задаються за допомогою виразу. Вираз може задаватися за зразком або використовувати складні функції вибору.

Наприклад, можна вибрати товари, що поставлені у поточному місяці постачальниками, розташованими у визначеному регіоні.

Зв'язування таблиць у запиті

Дуже часто виникає необхідність вибірки інформації з декількох таблиць.

> **Увага**
> Всі таблиці в запиті повинні бути пов'язані, принаймні, з одною таблицею. Якщо, наприклад, помістити в запит дві таблиці та не з'єднати їх, то Access, створить запит на основі **декартового добутку** цих двох таблиць (повний перебір).

Зв'язки між таблицями можна створити таким чином:

– при створенні макета бази даних;

– вибрати для запиту дві таблиці з однаковими полями одного типу, одне з яких також є **ключовим** полем в одній з цих таблиць;

– створити зв'язки у вікні конструктора запитів.

У перших двох способах зв'язки встановлюються автоматично. Проте іноді необхідно додати до запиту незв'язані таблиці, наприклад:

– дві таблиці мають загальне поле, яке назване різними іменами;

– таблиця не пов'язана (й не може бути пов'язана) з іншою таблицею.

Зв'язати дві таблиці, що не були автоматично пов'язані, можна у вікні конструктора запитів. Проте таке з'єднання не призводить до створення постійного зв'язку між цими таблицями. Такий зв'язок буде функціонувати тільки в запиті, з яким ви працюєте.

Типи зв’язків між таблицями

Між таблицями й запитами можна встановити такі типи зв'язків:

– один-до-одного;

– один-до-багатьох;

– багато-до-багатьох.

При завданні зв'язку між таблицями встановлюються тільки правила зв'язку, але не засіб перегляду даних на основі цього зв'язку.

Для перегляду даних у двох таблицях їх необхідно з'єднати загальним полем (або групою полів) в обох таблицях.

Такий метод з'єднання таблиць називається **з'єднанням** (**joining**). Існують різноманітні типи з'єднань:

– рівні з'єднання (внутрішні з'єднання);

– зовнішні з'єднання;

Access дозволяє змінити тип з'єднання. Для цього треба вибрати лінію зв'язку та двічі клікнути на ній лівою кнопкою миші. На екрані з'явиться діалогове вікно **Параметри з'єднання** (рис. 2.2).

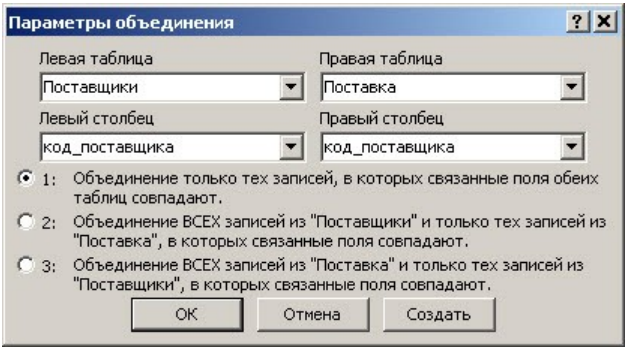


Рисунок 2.2 - Діалогове вікно «Параметри з'єднання»

У вікні є три перемикачі. Перший варіант зазвичай називають **внутрішнім** (INNER JOIN – у термінах мови SQL) **з'єднанням**, а два інших – зовнішнім.

Внутрішнє з'єднання (рівне з'єднання) прийнято в Access по умовчанню. Означає, що з таблиць вибираються тільки записи, що мають однакове значення в обох таблицях.

Зовнішні з'єднання відображають усі записи, що мають відповідні друг другу дані в обох полях. Вони також показують записи з однієї таблиці, що не мають підпорядкованих їм даних в іншій таблиці. Існує два види зовнішніх з'єднань: **ліві** (**LEFT JOIN**) і **праві** (**RIGHT JOIN**).

> **Примітка**
> Слід мати на увазі, що при перекладі мови інтерфейсу Access термін **Join** (операція реляційної алгебри **з'єднання**) був помилково перекладений як **об'єднання** (а це вже принципово інша операція – **Union**). Ця прикра помилка також дуже часто зустрічається у технічній літературі.

2.3.2 Завдання умов відбору записів

Оператор Like і символи підстановки

У полі **Умови відбору** бланка запиту можна задавати маску відбору за допомогою оператора **Like**, у якому використовуються символи підстановки (табл. 2.1 і табл. 2.2).

Таблиця 2.1 - Символи підстановки, які використовуються оператором Like

Спеціальний символ	Призначення
?	Будь-який символ (0-9, Аа-Zz, Аа-Яя)
*	Будь-яка кількість символів (0-n)
#	Будь-яка цифра

Таблиця 2.2 – Приклади використання символів підстановки разом з оператором Like

Вираз	У яких полях використане	Результат
Like "34*"	Клієнти.Телефон	Знаходить усі записи про клієнтів, телефони яких починаються на 34.
Like "[А-М]*"	Клієнти.Прізвища	Знаходить прізвища клієнтів, що починаються на літери від А до М
Like "*ОЗП*"	Товар.Опис	Знаходить усі записи, що містять слово ОЗП у будь-якому місці поля Опис.
Like "*.01.03"	Замовлення.Дата	Знаходить усі записи за січень 2003 року.
Like "!Монітор"	Товар.Опис	Усі товари, що відрізняються від "Монітор

> **Увага**
> Оператор **Like** і символи підстановки можуть використовуватися тільки для трьох типів полів: текстового, **Memo** та дати. При роботі з іншими типами полів це може призвести до помилки.

Завдання незбіжних значень

Для завдання незбіжних значень використовуються оператори **NOT** або оператор «не дорівнює» (<>). Вони записуються перед тим виразом, у якому записаний тип розбіжності.

Наприклад, **NOT Ріо-де-Жанейро** або **<> фонтан**.

Запровадження декількох умов відбору записів у одному полі

Для створення складних умов відбору записів можна використовувати декілька операторів. У основному такі умови відбору записів складаються з декількох операторів **AND** і **OR**. Для пошуку записів, що не задовольняють деякому значенню, можна використовувати оператор **NOT** із цим значенням.

При роботі з запитами ім'я поля треба брати в квадратні дужки, як при створенні полів, які обчислюються, або вказівці умов відбору записів, наприклад, [Дата Візиту]+30.

> **Увага**
> Якщо при вводі імені поля в умови відбору записів не включити його в квадратні скобки, Access автоматично візьме це ім'я в лапки й буде сприймати його, як текст, а не як ім'я поля.

Функції оператора OR

Оператор **OR** можна використовувати або в одній чарунці поля бланка запиту (рис. 2.3), або використовуючи рядок «**або»** (рис. 2.4).

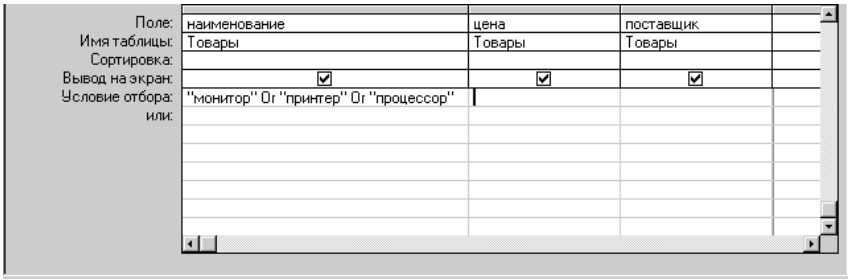


Рисунок 2.3 - Використання оператора OR в умовах відбору записів

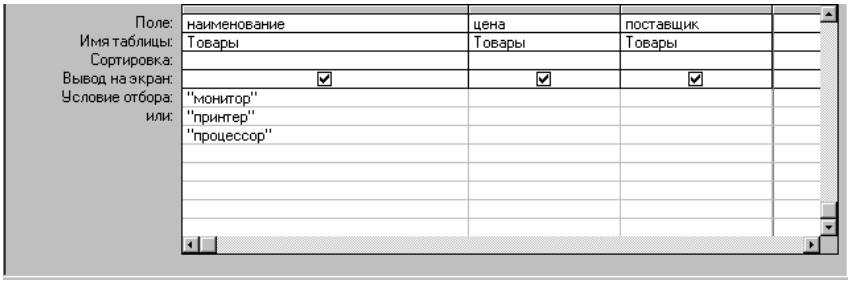


Рисунок 2.4 - Використання рядка або (OR)бланка запиту

Увага
Крім рядків **Умова відбору** та **або** (**OR**)у Access передбачено ще п'ять додаткових рядків **або** (**OR**) для запровадження умов відбору записів. При необхідності запровадження більшої кількості умов, можна використовувати оператор **OR** між окремими примірниками даних.

Використання списку значень і оператора IN

Для створення умов відбору для одного поля з використанням декількох примірників даних можна скористатися оператором **IN**. При виконанні запиту будуть відображені записи, що містять у зазначеному полі значення, що перераховані в списку оператора **IN.** Наприклад, **IN** («принтер»; «монітор») для поля **Найменування.**

Завдання діапазону значень

Діапазон значень можна задати за допомогою операторів **AND** або **BETWEEN…AND**...

Оператор **AND** використовується в тих запитах, у яких значення поля повинно одночасно задовольняти декільком умовам.

Наприклад, **>=100 And <=200** або **Between 100 And 200.**

Запровадження умов відбору записів за декількома полями

Для використання операторів **AND** і **OR** у запиті за декількома полями треба погоджувати розміщення примірників даних (або масок) в осередках **Умова відбору** й **або** (**OR**) одного поля з їхнім розміщенням в іншому полі.

Якщо ви хочете зв'язати декілька полів за допомогою оператора **AND,** розміщайте примірники даних (або маски) в одному й тому ж рядку. Для зв'язку умов оператором **OR** їх слід розміщувати в різних рядках бланка запиту (рис. 2.5).

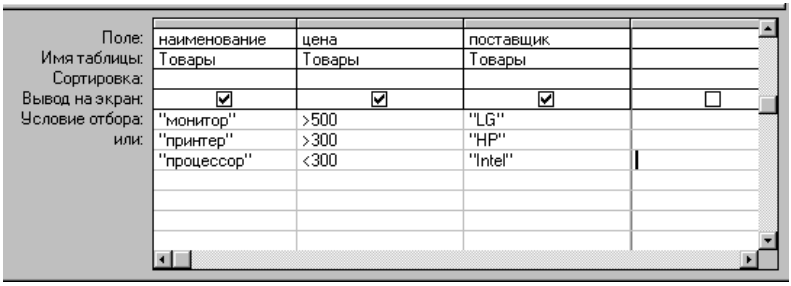


Рисунок 2.5 - Приклад використання операторів AND і OR у запиті

У даному прикладі умова задається формулою:

(«монітор» **And** >500 **And** «LG») **Or**
(«принтер» **And** >300 **And** «HP») **Or**
(«процесор» **And** <300 **And** «Intel»)

Створення в запиті поля, що обчислюється

При створенні запиту можна використовувати не тільки поля з таблиць, але й **поля, що обчислюються**. Наприклад, можна створити обчислюване поле **Премія**, у якому буде відображатися результат множення значення поля **Оклад** на значення поля **Кв_Премія**. Для цього у відповідну чарунку (осередок) бланка запиту потрібно ввести оператор **Премія: [Оклад] * [Кв_Премія]**.

> **Примітка**
> Ім'я для поля, що обчислюється, рекомендується задавати з двох основних причин: по-перше, для заголовка стовпчика таблиці запиту, та, по-друге, для звертання до цього поля у формі, звіті або іншому запиті.

2.2.3 Використання мови SQL для проектування запитів

Загальний формат команди SELECT

Для створення запитів використовується конструкція ***SELECT***. Результатом виконання запиту є таблиця, що зберігається в тимчасовому буфері серверу бази даних. Обрані дані можна використовувати для перегляду, друку звітів, формування графіків або, наприклад, додати їх до постійних (базових) таблиць бази даних.

Найбільш проста конструкція **SELECT** така:

SELECT список полів, що вибираються
FROM список таблиць;

Результатом запиту є інша таблиця, що утворюється із заданих у запиті таблиць. Наприклад:

```
select код, назва, ціна
from товари;
```

Якщо необхідно вивести всі стовпчики таблиці, то використовується символ «*»:

```
Select *
from товари;
```

Для виконання більш складних вибірок використовується загальна форма конструкції ***select:***

SELECT [***ALL***|***DISTINCT***] список полів, що вибираються
FROM список таблиць
[***WHERE*** умова вибірки]
[***GROUP BY*** умова угруповання
[***HAVIHG*** умова вибірки групи]]
[***ORDER BY*** умова впорядкування];

DISTINCT – аргумент, який усуває дублювання значень із результату виконання речення **SELECT.** Наприклад:

```
select distinct прізвище
from продавці;
```

Якщо оператор **SELECT** витягає декілька полів, то **DISTINCT** виключає рядки, в яких всі вибрані поля ідентичні, тобто він діє на всю строку, а не на окремі поля.

Альтернативою **DISTINCT** є ключове слово **ALL**, що має протилежне значення, тобто дозволяє виведення повторюваних рядків у запиті. По умовчанню використовується **ALL**.

При роботі з декількома таблицями, що містять однойменні поля, перед іменами стовпчиків треба задавати імена відповідних таблиць. У більшості діалектів **SQL** для спрощення набору таблицям дозволяється задавати **псевдоніми (aliases*)***. Псевдонім указується після імені таблиці в списку таблиць:

```
select T.код, T.назва, T.ціна
from Товари [as] T;
```

Буква ***T*** перед іменами стовпчиків заміняє повне ім'я таблиці (***товари***). Цей запит еквівалентний наступному:

```
select Товари.код, Товари.назва, Товари.ціна
from Товари;
```

Вибір рядків: речення WHERE

Речення **WHERE** задає предикат, умову, що може бути вірною або помилковою для кожного рядку таблиці. В результаті вибираються тільки ті рядки з таблиці, для яких предикат має значення «істина».

У **SQL** є ряд операторів і ключових слів для завдання умов (табл. 2.3).

Таблиця 2.3 – Оператори завдання умов

Оператор, ключове слово	Приклад використання
оператори порівняння (=, <, >, >=, <=, <>,*!* =);	where ціна > 1000;
комбінації умовних і логічних операцій – (*AND,OR,NOT*)	where ціна < 5000 and ціна >2000
діапазони (*BETWEEN* і *NOT BETWEEN*)	where ціна between 450 and 500
списки (*IN, NOT IN)*	where товар in («монітор», «принтер»)
невідомі значення (*IS NULL* і *IS NOT NULL)*	where телефон is null
відповідності символів *(LIKE* і *NOT LIKE)*	where телефон not like «415%»; where назва like «*БД*»

Розглянемо приклади.

Оператори порівняння

```
select код, назва, місто
from Постачальники
where місто =«Київ»;

select прізвище, посада
from Посадовці
where прізвище > «Іваненко»;

select *
from Продавці
where рейтинг > 100;

select *
from titles
where advance*2 > price + sales;
```

Комбінації умовних і логічних операцій

Логічні оператори виконуються в суворо визначеній послідовності (рис. 2.6). Якщо у виразі присутні обидва типи операторів, першими здійснюються арифметичні оператори.

Рисунок 2.6 – Послідовність виконання операцій

Роздивимося пошук моніторів в таблиці ***Товари***, незалежно від ціни, а також принтерів із ціною більше 700 грн.

Select модель, тип, ціна
From Товари
Where тип=«монітор» or тип=«принтер» and ціна >700;

Обмеження на ціну застосовується тільки до принтерів, тому що оператор ***AND*** виконується перед оператором ***OR***.

Щоб першим виконувався оператор ***OR***, у запиті треба

використовувати дужки. Наприклад, якщо потрібно вибрати всі принтери та сканери з ціною вище 1000 грн:

Select модель, тип, ціна
From Товари
Where (тип=«монітор» or тип=«принтер») and ціна >1000;

Роздивимося запит, що здійснює пошук книг, витрати на який не окупилися, тобто прибуток від продажу котрих менше подвоєної суми, що сплачена авторам. Крім того, введемо контрольну дату.

select назва, тип, ціна, гонорар, продано, дата
from Книги
where ціна*продано < 2* гонорар and дата < '15/09/20';

Діапазони (BETWEEN і NOT BETWEEN)

Діапазон можна визначити двома способами:

– за допомогою операторів порівняння <, >;

– за допомогою ключового слова (оператора) ***BETWEEN***.

Ключове слово ***BETWEEN*** можна використовувати для створення включаючого діапазону, якщо в результуючі значення повинні включатися й межі діапазону. Наприклад:

select модель, кількість
from Товари
where кількість between 100 and 150;

При використанні конструкції ***not between*** знаходяться рядки, значення яких лежать поза вказаним діапазоном. Наприклад:

select модель, кількість
from Товари
where кількість not between 100 and 150;

Аналогічний результат дає використання операторів порівняння:

select модель, кількість
from Товари
where кількість < 100 and кількість > 150;

Списки (IN, NOT IN)

Ключове слово **IN** дозволяє вибрати значення, що збігаються зі значеннями з заданого списку. Наприклад:

```
select name, royalty
from authors
where name in ('Довлатов', 'Бродський', 'Шмельов');
```

що еквівалентно запиту:

```
select name, royalty
from authors
where name='Довлатов'or name='Бродський'
or name='Шмельов';
```

Вибірка нульових (невідомих) значень

Значення ***NULL*** використовується для подання невідомої інформації. При цьому ***NULL***-значення не є звичайним нулем або порожньою позицією. Його не можна порівнювати з будь-яким іншим значенням. Проте, якщо за логікою запиту необхідно вибрати саме таку інформацію, то потрібно використовувати наступну конструкцію:

```
select код, картина, художник
 from Живопис
 where художник is null;
```

Пошук за шаблонами (LIKE і NOT LIKE)

При необхідності вибору інформації за неповними даними використовується така конструкція:

WHERE ім'я_стовпчика [NOT] LIKE 'зразок' [ESCAPE ключовий символ]

Ключове слово ***LIKE*** може бути застосовано до символьних полів і в деяких системах до полів дати, проте для чисельних полів воно не може бути застосовано.

Зразок (в лапках) може містити один або декілька шаблонів –

символів, які заміщають у зразку пропущені букви або рядки. Ключове слово ***ESCAPE*** використовується, якщо в зразку міститься один із шаблонів, але його потрібно розглядати як просту літеру.

У ***ANSI SQL*** використовуються два символи шаблону разом із ключовим словом ***LIKE:*** знак відсотка (%) і підкреслення (_).

% – будь-який рядок з будь-якою кількістю символів;

_ – будь-який одиночний символ.

Наприклад:

```
select автор, назва
 from Книги
       where назва like «%Access%»;

select *
 from Автори
   where прізвище like ‘Mc%’ or прізвище like ‘Mac%;

select *
 from Терміни
      where термін like «*»&[Введіть назву терміна]& «*»;
```

Щоб знайти в рядку символ підкреслення або відсотка, у предикаті ***LIKE*** будь-який символ можна визначити як ***ESCAPE-символ*** (керуючий символ). Він використовується в предикаті безпосередньо перед символом шаблона й означає, що наступний за ним символ інтерпретується саме як звичайний символ, а не символ шаблона.

```
select notes
 from table
       where notes like «%/%% escape /»;
```

У результаті будуть обрані всі значення, що містять символ «%».

Приклади:

like «27%» − рядок, що починається з 27;
like «27@%» − 27%;
like «_n» − an, in, on тощо;
like «@_n» − *_n*.

Вибірка із декількох зв'язаних таблиць

В цьому випадку використовується реляційний оператор **JOIN**.

Наприклад, треба визначити, які покупці, коли та на яку суму зробили покупки минулого місяця.

```
select Клієнти.найменування, Продажі.Дата,
       Продажі.Сума
from Клієнти INNER JOIN Продажі ON Клієнти.код =
       Продажі.Клієнт
where month(date())- month(Продажі.Дата)=1;
```

Факт зв'язування таблиць можна задати інакше:

```
select Клієнти.найменування, Продажі.Дата,
       Продажі.Сума
from Клієнти, Продажі
where (Клієнти.код = Продажі.Клієнт) and (month(date())-
month(Продажі.Дата)=1);
```

Обидва варіанти еквівалентні, але з точки зору «читабельності» у другому варіанті речення **where,** призначене для завдання умов відбору даних, явно перевантажується, що може призвести до можливого виникнення помилок.

В наступному прикладі для визначення товарів, які продавалися вчора, треба зв'язати три таблиці: Продажі, Продано та Товари:

```
select distinct Товари.товар, Продажі.Дата, Продажі.Сума
from Товари INNER JOIN (Продажі INNER JOINПродан
       ON Продажі.Ном = Продано.КодЗамовл) ON
       Товари.код_товара = Продано.КодТовару
where ((((Date()-[Продажі].[Дата]))=1));
```

При будь-якій кількості таблиць зв'язування виконується попарно в декілька етапів: спочатку зв'язуються дві таблиці, потім результат зв'язується з наступною таблицею. Цей ланцюжок продовжується, доти не будуть зв'язані всі таблиці.

Параметричні запити

Параметричний запит – це запит, при виконанні якого користувачу пропонується ввести значення якогось параметра, з урахуванням котрого й виконується запит.

Наприклад, створений запит, що відображає інформацію про кількість наявного товару на складі. Якщо цей запит використовується часто, то можна створити параметричний запит, який запитував би користувача про назву товару.

select Товари.найменування, Товари.Кількість,
from Товари
where Товари.найменування = [Введіть назву товару];

> 🕮 **Порада**
> Ви можете створити параметричні запити з використанням будь-якого припустимого оператора, включаючи оператор **Like** із шаблонами. Наприклад, параметр **Like [уведіть назву товару або <Enter>]&*** дозволяє користувачу запустити запит для одного товару або для всіх.

Аналогічно створюються багатопараметричні запити. При цьому варто мати на увазі, що вводити потрібно коректні значення, інакше в результуючій таблиці не виявиться жодного запису.

При використанні складного параметричного запиту може виникнути ситуація, коли параметри використовуються неодноразово, тому їх потрібно повторно вводити. Для запобігання цьому в SQL існує оператор **PARAMETERS**.

Оператор PARAMETERS служить для одноразового вводу значень окремих параметрів до виконання запиту, запам'ятовування їх та багаторазового використання без повторного введення.

Синтаксис оператора передбачає присутність імен параметрів (ідентифікаторів) та їх типу:

PARAMETERS <ідентифікатор> <тип даних> […, <ідентифікатор> <тип даних>];

Наприклад, необхідно визначити товари та суми їх продажу за вказаний період часу:

```
PARAMETERS [початок періоду] date, [кінець періоду] date;
select distinct Товари.товар, Продажі.Дата, Продажі.Сума
from Товари INNER JOIN (Продажі INNER JOINПродан
      ON Продажі.Ном = Продано.КодЗамовл) ON
      Товари.код_товара = Продано.КодТовару
where Продажі].[Дата] between [початок періоду] and [кінець періоду];
```

Оператор PARAMETERS доцільно використовувати при створенні запитів, на підставі яких проектуються звіти (лабораторна робота №5).

Контрольні питання

2.1 Що таке «запит»?
2.2 Які типи запитів підтримує Access?
2.3 Призначення і склад бланка запитів.
2.4 Вставка і вилучання полів із бланку запиту.
2.5 Як відсортувати таблицю по окремих полях?
2.6 Яким чином задаються критерії відбору записів?
2.7 Зв'язування таблиць у запиті.
2.8 Які існують типи зв'язків між таблицями
2.9 Які існують типи з'єднань між таблицями?
2.10 Що називається внутрішнім з'єднанням?
2.11 Що називається зовнішнім з'єднанням?
2.12 Що таке «віртуальна таблиця» та її відмінності від базової.
2.13 Використання функцій OR і AND при завданні умов відбору записів.
2.14 Призначення оператора Like?
2.15 Призначення оператора IN?
2.16 Завдання діапазону значень.
2.17 Як створити в запиті поле, що обчислюються?
2.18 Призначення та синтаксис оператора PARAMETERS?

3 ЛАБОРАТОРНА РОБОТА № 3 «СТВОРЕННЯ СКЛАДНИХ ЗАПИТІВ»

3.1 Мета роботи

Метою роботи є ознайомлення з основними прийомами проектування складних запитів.

3.2 Завдання до лабораторної роботи

3.2.1 Ознайомитися зі змістом пункту 3.3 методичних вказівок.

3.2.2 Для бази даних, створеної в попередніх лабораторних роботах, за узгодженням із викладачем розробити різноманітні типи запитів, використовуючи можливі варіанти їхнього створення.

3.3 Основні теоретичні відомості

3.3.1 Агрегатні функції

Агрегатні функції (табл. 3.1) використовуються для одержання узагальнюючих значень. Вони дають єдине значення для цілої групи рядків таблиці. Агрегатні функції завжди мають аргументи, що є виразами та містяться у дужках. У якості аргументів зазвичай використовуються назви стовпчиків, але також припускаються константи, функції та будь-які їхні комбінації з арифметичними операторами.

Таблиці 3.1 – Агрегатні функції

Функція	Результат
COUNT	Визначає кількість рядків або значень поля, обраних за допомогою запиту, що ***не є NULL-значеннями***
SUM	Обчислює арифметичну суму всіх обраних значень даного поля
AVG	Обчислює середнє значення для всіх обраних значень даного поля
MAX	Обчислює найбільше зі всіх обраних значень даного поля
MIN	Обчислює найменше зі всіх обраних значень даного поля

Для ***SUM*** і ***AVG*** можуть використовуватися тільки цифрові поля. Для **COUNT**, **MAX** і **MIN** – цифрові та символьні поля. Стосовно до символьних полів **MAX** і **MIN** визначають значення відповідно коду **ASCII** згідно алфавіту.

Функції **COUNT(DISTINCT)** і **COUNT(*)** відрізняються тим, що перша рахує кількість непорожніх і неповторюваних рядків, а друга – загальну кількість рядків.

Наприклад:

```
select count (*)
from Покупці;
```

В агрегатних функціях для обмеження кількості рядків, що беруть участь в обчисленнях, можна використовувати конструкцію **WHERE.**

Наприклад:

```
select назва, sum(сума)
from Продажі
where назва = «Монітор»
```

3.3.2 Групування даних

Речення **GROUP BY** розділяє таблицю на набори, а агрегатна функція обчислює для кожного з них підсумкове значення. Ці значення називаються **агрегатним вектором.**

```
SELECT список вибору
  FROM список таблиць
  [WHERE умови]
  [GROUP BY список_угруповання];
```

Наприклад:

```
select художник, count(картина)
 from Музеї
       group by художник;
```

До списку вибору включаються стовпчик, за яким проводиться угруповання, та агрегатна функція.

Спільна робота речень ***WHERE*** і ***GROUP BY*** відбувається в такий спосіб. Спочатку знаходяться всі рядки, що відповідають умові ***WHERE***. Потім речення ***GROUP BY*** поділяє відібрані рядки на групи. Рядки, що не задовольняють умовам речення ***WHERE***, не включаються до жодної групи.

```
select музей, count(картина)
  from Музеї
  where художник =”Рубенс”
        group by музей;
```

Речення HAVING

Якщо в списку вибору є агрегатні функції, речення **WHERE** виконується перед ними, тоді як речення **HAVING** застосовується до всього запиту в цілому, після розбивки на групи та обчислення значень функцій.

З погляду синтаксису речення **WHERE** і **HAVING** ідентичні. Відмінність полягає у тому, що в умові речення **WHERE** не можуть знаходитися агрегатні функції. Крім того, у більшості систем елементи речення **HAVING** повинні включатися до списку вибору. На речення **WHERE** це обмеження не поширюється.

Пропозиція **HAVING** працює наступним чином: спочатку **GROUP BY** розділяє рядки на групи, для яких обчислюється агрегатна функція, потім на отримані результати накладаються умови речення **HAVING:**

```
select тип, count(*)
 from Товари
 group by тип
 having count(*) > 1;
```

3.3.3 Створення перехресних запитів

Щоб полегшити сприйняття зведених даних в Access, варто скористатися перехресним запитом. Перехресний запит обчислює суму, середнє значення або іншу агрегатну функцію, а потім групує результати за двома наборами значень: збоку таблиці даних і в її верхній частині Цей запит представляє визначені дані з обраних полів у форматі, схожому на формат електронної таблиці.

Для створення перехресного запиту необхідно визначити, як мінімум, три параметри:

– поле заголовків рядків;
– поле заголовків стовпчиків;
– поле для вибору значень.

Загальний формат перехресного запиту має такий вид:

TRANSFORM <Агрегатна функція(Ім'я поля для вибору значень**) >**
SELECT <Ім'я поля заголовків рядків**>**
FROM <Зв'язані таблиці для вибірки>
GROUP BY <Ім'я поля заголовків рядків**>**
PIVOT <Ім'я поля заголовків стовпчиків**>**;

Припустимо, в складі бази даних є зв'язані таблиці:
- **Продажі** (Код, Дата, Сума);
- **Продано** (НомЗап, НомПрод, КодТовару, Ціна, Кількість, Разом);
- **Товари** (Код, Група, Товар, ЦінаЗак, ЦінаВідп).

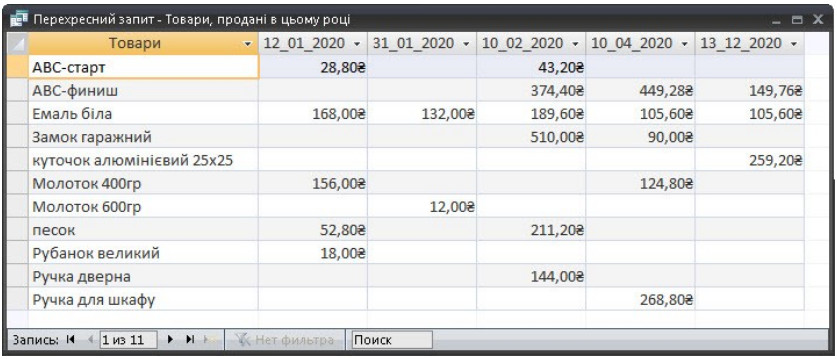
Необхідно створити запит, який відображав би в результуючій таблиці товари як заголовки рядків, дати продажу як заголовки стовпчиків, а в кожній чарунці таблиці містилися б дані про суму, на яку проданий конкретний товар даного числа.

```
transform Sum(Продано.Разом)
select Товари.товар AS [Товари]
        from Продажі INNER JOIN (Товари INNER JOIN
        Продано ON Товари.Код =
        Продано.КодТовару) ON Продажі.Ном =
        Продано.НомПрод
        group by Товари.товар
pivot Продажі.Дата;
```

Ускладнимо запит, додавши період часу, за який треба зробити вибірку.

```
parameters початок date, кінець date;
transform Sum(Продано.Разом)
select Товари.товар AS [Товари]
        from Продажі INNER JOIN (Товари INNER JOIN
        Продано ON Товари.код_товара = Продано.КодТовару)
        ON Продажі.Ном = Продано.КодЗамовл
        where Продажі.Дата Between початок and кінець
        group by Товари.товар
pivot Продажі.Дата;
```

Результат виконання запиту показаний на рисунку 3.1.



Перехресний запит - Товари, продані в цьому році

Товари	12_01_2020	31_01_2020	10_02_2020	10_04_2020	13_12_2020
АВС-старт	28,80₴		43,20₴		
АВС-финиш			374,40₴	449,28₴	149,76₴
Емаль біла	168,00₴	132,00₴	189,60₴	105,60₴	105,60₴
Замок гаражний			510,00₴	90,00₴	
куточок алюмінієвий 25х25					259,20₴
Молоток 400гр	156,00₴			124,80₴	
Молоток 600гр		12,00₴			
песок	52,80₴		211,20₴		
Рубанок великий	18,00₴				
Ручка дверна			144,00₴		
Ручка для шкафу				268,80₴	

Запись: 1 из 11 | Нет фильтра | Поиск

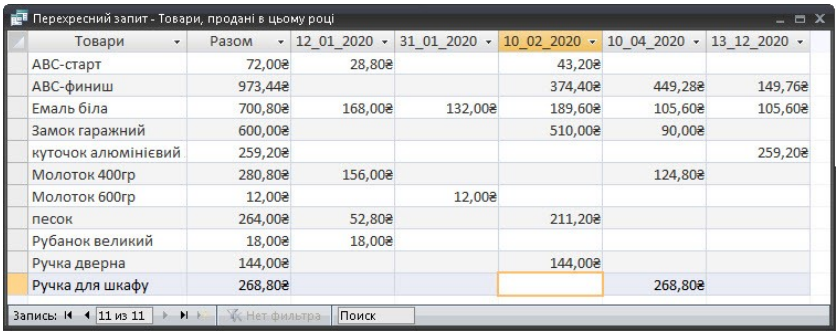
Рисунок 3.1 - Результат виконання перехресного запиту

Перехресний запит може мати декілька полів для заголовків рядків. Наприклад, в попередньому запиті можна додати підсумок продаж по кожному товару:

```
parameters початок date, кінець date;
transform Sum(Продано.Разом) AS Усі
select Товари.товар AS [Товари], Sum(Усі) AS [Разом]
        from Продажі INNER JOIN (Товари INNER JOIN
        Продано ON Товари.код_товара =
        Продано.КодТовару)
        ON Продажі.Ном = Продано.КодЗамовл
        where Продажі.Дата Between початок and кінець
        group by Товари.товар
pivot Продажі.Дата;
```

Результат виконання запиту показаний на рисунку 3.2.

> 🕮 **Порада**
> Перехресний запит може мати декілька заголовків рядків і тільки один заголовок стовпчиків. Якщо необхідно створити запит з одним заголовком рядків і декількома заголовками стовпчиків, поміняйте місцями рядки та стовпчики.


Перехресний запит - Товари, продані в цьому році

Товари	Разом	12_01_2020	31_01_2020	10_02_2020	10_04_2020	13_12_2020
АВС-старт	72,00₴	28,80₴		43,20₴		
АВС-финиш	973,44₴			374,40₴	449,28₴	149,76₴
Емаль біла	700,80₴	168,00₴	132,00₴	189,60₴	105,60₴	105,60₴
Замок гаражний	600,00₴			510,00₴	90,00₴	
куточок алюмінієвий	259,20₴					259,20₴
Молоток 400гр	280,80₴	156,00₴			124,80₴	
Молоток 600гр	12,00₴		12,00₴			
песок	264,00₴	52,80₴		211,20₴		
Рубанок великий	18,00₴	18,00₴				
Ручка дверна	144,00₴			144,00₴		
Ручка для шкафу	268,80₴				268,80₴	

Запись: 11 из 11 | Нет фильтра | Поиск

Рисунок 3.2 - Результат виконання перехресного запиту

3.3.4 Команди модифікації даних

У ***SQL*** використовуються три команди модифікації даних:

- ***INSERT*** додає нові рядки до таблиці:
- ***UPDATE*** змінює існуючі в таблиці рядки;
- ***DELETE*** вилучає рядки з таблиці.

> 🕮 **Порада** Завжди зберігайте резервну копію даних перед виконанням команди модифікації

Додавання нового рядку

Оператор ***INSERT*** дозволяє додавати рядки за допомогою ключового слова ***VALUES*** або за допомогою оператора ***SELECT***. Для кожного рядку, що додається, використовується свій оператор ***INSERT***.

```
INSERT INTO ім'я_таблиці [( стовпчик1 [, стовпчик2]…)]
VALUES( константа1 [, константа2]…)
```

Наприклад. треба додати новий товар до таблиці ***Товари(код, назва, ціна, кількість).***

```
insert into Товари (код, назв, ціна, кількість)
        values( '1756', «монітор», 2000, 6);
```

Порядок слідування стовпчиків в операторі ***INSERT*** може бути будь-яким, проте він повинний відповідати порядку слідування значень.

Якщо рядок, який додається, містить значення окремих стовпчиків таблиці, то стовпчики, що залишилися, (що не модифікуються) повинні припускати нульовий статус. У противному випадку команда буде відхилена.

Для додавання даних з однієї або декількох таблиць у команді ***INSERT*** можна використовувати оператор ***SELECT:***

```
INSERT INTO ім'я_таблиці [(список стовпчиків,
які вставляється)]
        SELECT список_стовпчиків
        FROM список_таблиць
                WHERE умови
```

Оператор ***SELECT*** у команді ***INSERT*** дозволяє взяти дані з декількох або з усіх стовпчиків однієї таблиці та вставити їх в іншу таблицю. Якщо вставлені значення тільки для частини стовпчиків, визначити значення для інших стовпчиків можна буде пізніше за допомогою оператора ***UPDATE***.

При вставці рядків з однієї таблиці в іншу ці таблиці повинні мати сумісну структуру. Якщо стовпчики в обох таблицях сумісні за типами та визначені в однаковому порядку у відповідних операторах ***CREATE TABLE***, перераховувати їх у команді ***INSERT*** необов'язково.

Наприклад, до таблиці ***Прайс-лист(тип, назва, ціна)*** треба додати інформацію з таблиці ***Товари***(***тип, назва, ціна_закуп, ціна_відпускна, кількість)***

```
insert into Прайс-лист(тип, назва, ціна)
        select тип, назва, ціна_відпускна
        from Товари
```

Оператор ***SELECT*** у команді ***INSERT*** може містити агрегатні функції. Наприклад, є таблиці ***Підсумки(дата, сума)*** та ***Продаж(ном, дата, вартість)***. Необхідно підрахувати щоденну виручку за минулий місяць.

```
insert into Підсумки (дата, сума)
    select дата, sum(вартість)
    from Продаж
    group by дата;
```

При використанні підзапитів у всіх командах оновлення в реченні ***FROM*** будь-якого підзапиту не можна посилатися на таблицю, що змінюється в основній команді.

Побудова запиту на створення таблиці

Оператор **Select … Into** дозволяє створювати нову таблицю на базі однієї або декількох інших таблиць. При цьому таблиця заповнюється даними з відповідних таблиць.

Увага При повторному запуску запиту створена раніше таблиця вилучається й замінюється новою таблицею.

Наприклад, необхідно створити таблицю **Прайслист** на базі існуючої таблиці **Товари (**Код, Група, Назва, Ціна_закупочна, Ціна_відпускна, Складські_запаси**).**

```
Select Код, Група, Назва, Ціна_закупочна, Ціна_відпускна As Ціна
Into Прайслист
From Товари;
```

Примітка
При створенні таблиці за допомогою запиту типи даних і розміри полів зберігаються такими ж, якими вони були в базовій таблиці запиту. Проте ніякі інші властивості полів і таблиці не успадковують!

Зміна значень полів

При необхідності значення полів можна змінювати за допомогою операції оновлення ***UPDATE,*** яка має такий формат:

```
UPDATE ім'я таблиці
 SET ім'я стовпчика = нове_значення
 [WHERE умова_відбору]
```

При відсутності речення ***WHERE*** зміни вносяться у всі рядки до стовпчиків, зазначених у реченні ***SET***.

В таблиці **Співробітники** підвищити оклад на 25% тим

співробітникам, у яких стаж роботи більше 10 років.

```
UPDATE Співробітники
SET Оклад = Оклад*1,25
WHERE Year(date())-Year(Дата_прийому) >10;
```

Побудова запиту на вилучення записів

З усіх запитів на зміну запит на вилучення найбільше небезпечний. На відміну від інших запитів він вилучає записи раз і назавжди.

Запит на вилучання може одночасно вилучати записи з декількох таблиць. Проте для вилучання пов'язаних записів із декількох таблиць необхідно виконати такі дії:

– визначити зв'язки між таблицями у вікні **Схема даних**.

– встановити опцію **Забезпечення цілісності даних** для заданого типу зв'язку між таблицями.

– встановити опцію **Каскадне вилучання пов'язаних записів** для заданого типу зв'язку між таблицями.

При роботі зі зв'язками **один-до-багатьох** без завдання зв'язків і без умикання параметра **Каскадне вилучання пов'язаних записів** Access вилучить записи тільки з однієї таблиці. Порядок такий: Access спочатку вилучає записи з боку **багато** цих зв'язків, а потім вам належить вилучити записи з боку **один**.

> **Увага**
> Оскільки запит на вилучання робить необоротний руйнуючий вплив на таблиці, перед початком роботи з ними завжди варто виконувати архівне копіювання таблиць.

За допомогою команди ***DELETE*** із таблиці вилучається не окреме поле, а рядок цілком.

Синтаксис оператора такий:

```
DELETE
FROM ім'я_таблиці
[WHERE умови_відбору]
```

> **Примітка**
> Запит на вилучання цілком вилучає записи, а не тільки зазначені поля даних. Для вилучання даних з окремих полів необхідно використовувати запит на оновлення, щоб замінити значення цих полів на порожні значення (**Null**).

3.3.5 Запити на об'єднання UNION

Крім вкладення запитів їх можна також об'єднувати. ***Об'єднання*** (***unions***) відрізняються від підзапитів тим, що будь-який із двох (або більшого числа) запитів не може управляти іншим запитом.

Можна задати множину запитів одночасно та комбінувати їх результати з використанням речення ***UNION***, що об'єднує результати декількох SQL-запитів у єдину множину рядків і стовпчиків.

Наприклад, з таблиць ***Викладачі***, ***Співробітники*** та ***Аспіранти*** необхідно вибрати інформацію про дні народження та упорядкувати її за місяцями та числами:

```
Select Прізвище, дата_народж
        From Викладачі
Union
Select Прізвище, дата_народж
        From Співробітники
Union
Select Прізвище, дата_народження
        From Аспіранти
Order by month(дата_народж), day(дата_народж);
```

Стовпчики, обрані за допомогою трьох операторів ***SELECT***, подані у вихідних даних так, якби вони вибиралися за допомогою одного запиту. При цьому заголовки стовпчиків беруться з першого оператора ***SELECT***.

Примітки:

1. Для об'єднання двох і більш запитів необхідно, щоб їхні стовпчики, що входять у набір вихідних даних, були ***сумісні по об'єднанню.***

2. Якщо NULL-значення заборонені для будь-якого стовпчика в об'єднанні, то вони повинні бути заборонені для усіх відповідних стовпчиків в інших запитах об'єднання.

3. Кожен оператор ***SELECT,*** який входить до оператора ***UNION,*** може мати своє власне речення ***WHERE.***

4. У реченні сортування ***Order by*** назва атрибутів повинна братися з першого запиту і застосовуватися до всього результату.

5. Не можна використовувати ***UNION*** у підзапитах.

6. В операторі ***UNION*** не можна використовувати функції агрегування в реченнях ***SELECT.***

Оператор Union можна використати як різновид оператора IF для відображення різних значень для одного поля, залежно від значень в інших полях. Без оператора Union для одержання аналогічного результату потрібне виконання декількох запитів.

Нехай, наприклад, треба одержати список книг, указавши для кожної з них процентне зниження ціни та нову вартість. Вартість книг до 40 грн знижується на 10%, між 40грн і 50грн — на 20%, вище 50 грн — на 30%.

Без оператора Union треба було б виконати три окремих запита та помістити результати в нову таблицю.

Оператор Union виконує все це за один прохід:

```
Select "10%", Назва, Ціна As [Стара ціна], Ціна*0,9 As [Нова ціна]
From Книги
Where Ціна<40
Union
Select "20%", Назва, Ціна As [Стара ціна], Ціна*0,8 As [Нова ціна]
From Книги
Where Ціна>40 and Ціна<50
Union
Select "30%", Назва, Ціна As [Стара ціна], Ціна*0,7 As [Нова ціна]
From Книги
Where Ціна>50;
```

Контрольні питання

3.1 Поняття групового запиту.
3.2 Категорії групових запитів.
3.3 Що називається агрегатними функціями, для чого вони використовується?
3.4 Склад і призначення агрегатних функцій.
3.5 Визначення критеріїв для групового запиту.
3.6 Призначення операторів **Where** та **Having** .
3.7 Створення виразу для групових операцій.
3.8 Створення перехресного запиту.
3.9 Типи запитів на зміну.
3.10 Створення запитів на оновлення.
3.11 Створення запиту на створення таблиці **Select … Into**.
3.12 Створення запитів на додавання записів.
3.13 Побудова запиту на вилучання записів.
3.14 Запит Union.

РЕКОМЕНДОВАНА ЛІТЕРАТУРА

1. Корнієнко С. К. Проектування інформаційного забезпечення автоматизованих систем: Навч. Посібник / С.К. Корнієнко . – Запоріжжя: ЗНТУ, 2015. – 224 с.

2. Шпортько А. Розробка баз даних в СУБД Microsoft Access / А. Шпортько, Л. Шпортько. – Київ: Кондор, 2018. – 184 с.

3. Завадський І.О. Основи баз даних / І.О. Завадський. – Київ: ПП І.О. Завадський, 2011. –192 с.

4. Морзе Н.В. Бази даних у навчальному процесі / Н.В. Морзе. – Київ: ТОВ Редакція «Комп'ютер», 2007. – 120 с.

5. Дунаев В. В. Базы данных. Язык SQL для студента /В. В. Дунаев – СПб.: «БХВ-Петербург», 2007/ – 234 с,

6. Бекаревич Ю. Microsoft Access 2016 / Ю. Бекаревич, Н. Пушкина. – СПб.: «БХВ-Петербург», 2016. – 408 с.

7. Бекаревич Ю. Microsoft Access 2013 / Ю. Бекаревич, Н. Пушкина. – СПб.: «БХВ-Петербург», 2013. – 465 с.

8. Бекаревич Ю. Microsoft Access 2010 / Ю. Бекаревич, Н. Пушкина. – СПб.: «БХВ-Петербург», 2011. – 432 с.

9. Гурвиц Г. Microsoft Access 2010. Разработка приложений на реальном примере / Г. Гурвиц . – СПб.: «БХВ-Петербург», 2010. – 496 с.