Міністерство освіти і науки України
Миколаївський національний університет
імені В.О.Сухомлинського

Кафедра прикладної математики та
інформаційних комп'ютерних технологій

Г.С. Погромська

ПОБУДОВА ЗАПИТІВ НА МОВІ SQL

Навчальний посібник
з курсу «Бази даних та інформаційні системи»

для студентів напряму підготовки

0403 «Системні науки та кібернетика»

Миколаїв
2014

УДК 004.655(076.5)
ББК 32.81
П 43

Рекомендовано до друку кафедрою прикладної математики та інформаційних комп'ютерних технологій Миколаївського національного університету імені В.О.Сухомлинського (протокол № 4 від 03.10.2014 р.)

Рекомендовано до друку рішенням Вченої ради Миколаївського національного університету імені В.О.Сухомлинського (протокол №__ від _____)

Рецензенти:

П 43
Погромська Г.С. Побудова запитів на мові SQL: Навчальний посібник. – Миколаїв: _____ – 133 с.

Навчальний посібник вміщує теоретичний і практичний матеріал з вивчення SQL – мови запитів до баз даних. На практичних прикладах докладно описані основні конструкції мови, а також різні типи запитів: прості, складені тощо. Розглянуто основи реляційної алгебри, запити за допомогою агрегатних функцій, операції над наборами записів, з'єднання таблиць, модифікації даних, створення уявлень.

Для студентів і викладачів з напряму «Системні науки та кібернетика» вищих навчальних закладів на підтримку вивчення курсу «Бази даних та інформаційні системи», а також усіх, хто має намір оволодіти засадами роботи з вивчення основ мови маніпулювання даними – SQL у середовищі СУБД MS Access.

УДК 004.655(076.5)
ББК 32.81

ЗМІСТ

ПЕРЕДМОВА

Інформатизація суспільства зростає швидкими темпами, компанії, офіси, підприємства, навчальні та наукові заклади все ширше використовують інформаційні технології в своїй діяльності. В більшості випадків ядром, навколо якого будується інформаційна інфраструктура організації , виступають бази даних, як правило, реляційного типу. Реляційна модель, яка домінує над іншими моделями даних, тісно пов'язана з мовою SQL (Structured Query Language).

Фактично, на даний час, різних СУБД реляційного типу існує багато, а універсальний засіб робот з ними єдиний – SQL (широко розповсюджена базова мова роботи з реляційнмими базами даних). Останнє робить актуальним вивчення мови SQL як у практичному плані, та і суто теоретично, оскільки в основі елементів мови SQL лежать теорії відношень, теорії множин та логіки. Знання засад SQL та вміння їх застосовувати для пошуку та аналізу даних є фундаментальною частиною інформатичної компетентності кваліфікованих програмістів.

Модуль «Мова SQL» є складовою бакалаврського курсу «Бази даних та інформаційні системи» спеціальності «Інформатика» і має за мету забезпечити можливості вивчення теоретичних основ та практичного застосування мови SQL під час роботи з реляційними базами даних.

Метою пропонованого посібника є допомогти студенту раціонально організувати самостійну роботу під час вивчення курсу «Бази даних та інформаційні системи», сприяти формуванню інформаційно-технологічних умінь студентів під час роботи з об'єктами баз даних.

Після опрацювання матеріалу посібника студент повинен *вміти* мовою SQL:

- створювати таблиці;
- отримувати дані з таблиць;
- застосовувати функції агрегуванні при формуванні звітів;
- створювати запити з декількох таблиць;
- створювати підзапити;
- виконувати самооб'єднання таблиць;
- працювати з уявленнями;
- видаляти об'єкти бази даних.

Навчально-методичний посібник складається з п'яти частин: теоретичні відомості, лабораторний практикум, додаткові завдання для індивідуального виконання, тестові питання для самоконтролю, варіант контрольної роботи. У кінці посібника наведена література за розділами: базова, допоміжна, інформаційні ресурси, термінологічний словник, умовні скорочення та додатки.

Теоретичні відомості складаються з 20-ти пунктів і розкривають базові положення з: операцій реляційної алгебри (основних та спеціальних операції): об'єднання, перетин, різниця, добуток, вибірка, з'єднання та ділення; історії створення мови SQL, складу мови, створення таблиць мовою SQL, роботи з оператором SELECT; необхідності використання функцій

агрегування, групування даних; аспекти застосування вкладених запитів (підзапитів), об'єднання запитів та об'єднання таблиць мовою SQL; видів запитів (перехресні, з параметрами, на створення таблиць, на додавання, видалення та поновлення (як частина модифікації даних засобами DML)). Надається визначення уявлень та видалення об'єктів БД. Виклад матеріалу з кожного пункту ілюструється великою кількістю прикладів.

Загальна схема вивчення теоретичних відомостей пропонується наступна: вводяться теоретичні положення, ставиться змістовий приклад, пропонується запит на мові SQL, подається результат. Надалі потрібно з використанням інструментальних засобів роботи з SQL проробити запропонований запит на мові SQL та впевнитися в правильності його отримання. За необхідності пояснюються конструкції мови SQL та робляться інші зауваження навчально-методичного характеру, які сприяють засвоєння матеріалу.

Лабораторний практикум спрямований на закріплення теоретичних положень з курсу. Перелік лабораторних робіт складається з 4-х тем. Лабораторна робота №1 присвячена введенню в мову SQL та синтаксису оператора SELECT. Лабораторна робота № 2 розкриває відомості з функцій агрегування та механізму групування даних. Лабораторна робота № 3 відноситься до тематики підзапитів, об'єднання запитів та таблиць. Лабораторна робота № 4 пропонує відомості зі створення перехресних запитів та операцій з модифікації даних та об'єктів БД засобами DML, роботи з уявленнями. Кожна лабораторна робота подана в єдиному форматі і містить такі розділи: тема, мета, вимоги до звіту, завдання для самостійної роботи, контрольні питання.

Лабораторний практикум поданий у вигляді набору згрупованих за темами завдань, виконання яких дозволить отримати систематизовані уміння з формування запитів на мові SQL. Кожна тема лабораторної роботи пропонує послідовне вивчення конструкцій операторів мови SQL за мірою збільшення їх складності.

Під час підготовки до виконання робіт студент повинен опрацювати теоретичні відомості з відповідної теми, знайти відповіді на контрольні питання у поданому матеріалу або у рекомендованій літературі. Це допоможе в оформленні звітів до лабораторних робіт.

Для засвоєння вивченого матеріалу і самоконтролю знань пропонуються додаткові завдання для індивідуального виконання, тестові завдання та орієнтовний варіант контрольної роботи.

1. ТЕОРЕТИЧНІ ВІДОМОСТІ

1.1. Реляційна алгебра

1.1.1. Поняття реляційної алгебри

Основним компонентом тієї *частини реляційної моделі*, яка стосується операторів, є так звана **реляційна алгебра,** яка у більшій частині *складається з набора операторів, які використовують відношення в якості операторів та які повертають відношення в якості результата.*

Реляційна алгебра, визначена Е. Кодом, складається з восьми операторів, які складають дві групи (по чотити оператори в кожній):

1. *Традиційні операції над множинами:* об'єднання, перетин, різниця та декартовий добуток (модифіковані з урахуванням того, що їх операндами є відношення, а не довільні множини).
2. *Спеціальні реляційні операції:* вибірка, проекція, з'єднання та ділення.

1.1.2. Замкнутість в реляційній алгебрі

Результат кожної операції над відношенням (або реляційної операції) також є відношенням. Ця реляційна властивість називається *властивістю замкнутості*. Оскільки результат будь-якої операції має той же тип, що і вихідні об'єкти (відношення), то результат однієї операції може використовуватися в якості вихідних даних для іншої. Таким чином, є можливість, наприклад, взяти або проекцію від об'єднання, або з'єднання від двох вибірок, або об'єднання з'єднання та перетину і т. ін.

Тобто, можна записувати вкладені вирази, тобто вирази, в яких операнди самі подані виразами замість простих імен відношень.

Якщо розглядати замкненість більш строго, кожна реляційна операція повинна бути визначена таким чином, щоб видати результат з належним заголовком (тобто з відповідним набором потрібних імен атрибутів). Причина такої вимоги до результуючих відношень полягає в необхідності мати можливість звертатися до імен атрибутів в наступних операціях, наприклад в подальших операціях, які розташовані на більш глибших рівнях вкладеного виразу. Іншими словами, потрібно мати такий набір правил спадкування атрибутів, вбудований в алгебру, щоб можна було передбачити імена атрибутів на виході довільної реляційної операції, знаючи імена атрибутів на вході цієї операції.

1.1.3. Операції реляційної алгебри
Основні операції реляційної алгебри

В основі реляційної моделі є математичне поняття теоретико-множинного відношення. Теоретико-множинне відношення являє собою

підмножину декартова добутка доменів. **Доменом** називається набір значень елементів даних одного типу, який відповідає поставленим умовам (наприклад, домен *ПІБ* визначається на базовому типі рядків символів, але до числа його значень можуть входити тільки ті рядкі, які можуть відображати ім'я).

Декартовим добутком k доменів (D_1, D_2,…, D_k), (позначається $D_1 \times D_2 \times \ldots \times D_k$), називається множина усіх кортежів вида (V_1, V_2, …, V_k) довжини k, таких, що $V_1 \in D_1$, $V_2 \in D_2$, …, $V_k \in D_k$.

Наприклад, нехай $D_1 = \{1, 2, 3\}$; $D_2 = \{a, b, c, d\}$

Тоді

$$D_1 \times D_2 = \left\| \begin{array}{cccc} 1,a & 1,b & 1,c & 1,d \\ 2,a & 2,b & 2,c & 2,d \\ 3,a & 3,b & 3,c & 3,d \end{array} \right\| \leftarrow \begin{array}{l} \text{в} \\ \text{матричному} \\ \text{вигляді} \end{array}$$

або:

$D_1 \times D_2$ = {(1,a), (1,b), (1,c), (1,d), (2,a), (2,b), (2,c), (2,d), (3,a), (3,b), (3,c), (3,d)}

Відношенням називають деяку підмножину декартового добутку доменів (передбачаються тільки кінцеві відношення).

Приклади відношень:

1) {(1,a), (1,c), (1,b)} – підмножина декартова добутка доменів $D_1 \times D_2$;

2) Ø.

В реляційних СУБД для виконання операцій над відношеннями використовують дві групи мов, які мають в якості своєї математичної основи теоретичні мови запитів, запропоновані Е. Кодом.

онных СУБД для выполнения операций над отношениями используют две группы языков, имеющие в качестве своей математической основы теоретические языки запросов, предложенные Е. Коддом:

- реляційну алгебру;
- реляційне числення.

Реляційною алгеброю називають систему операцій маніпулювання відношеннями, кожен оператор якої в якості операнда має одне або більше відношень та утворює відношення за раніше обумовленим правилом.

Варіант реляційної алгебри, запропонований Е. Кодом, включає наступні операції: об'єднання, різниця, перетин, декартовий (прямий) добуток, вибірка (селекція), проекція, ділення та з'єднання. Спрощене графічне подання перерахованих операцій показано на рис. 1.

1) **Об'єднанням** двох відношень ($r \cup s$) є відношення, що утримує всі кортежі, які належать або r, або s, або обом відношенням.

Данна операція застосовується до відношень з однаковою схемою.

Схемой відношення r називається кінцева множина імен атрибутів $\{A_1, A_2, \ldots, A_n\}$.

Приклади.

Приклад1.1. Нехай k та m – відношення зі схемою ABC:

k	(A	B	C)
	a_1	b_1	c_1
	a_1	b_2	c_1
	a_2	b_1	c_2

m	(A	B	C)
	a_1	b_2	c_1
	a_2	b_2	c_1
	a_2	b_2	c_2

Тоді об’єднання даних відношень:

$k \cup m =$	(A	B	C)
	a_1	b_1	c_1
	a_1	b_2	c_1
	a_2	b_1	c_2
	a_2	b_2	c_1
	a_2	b_2	c_2

Приклад1.2. Нехай відношення R1 – множина постачальників з Миколаєва, а R2 – множина постачальників, які поставляють деталь D1:

R1

№_П	ПІБ_П	Місто_П
P1	Іванов І. І.	Миколаїв
P4	Петров П. П.	Миколаїв

R2

№_П	ПІБ_П	Місто_П
P1	Іванов І. І.	Миколаїв
P2	Сидоров С. С.	Київ

Тоді об’єднання даних відношень дає постачальників з Миколаєва, або постачальників, які поставляють деталь D1, або тих і інших.

R1 ∪ R2

№_П	ПІБ _П	Місто_П
P1	Іванов І. І.	Миколаїв
P4	Петров П. П.	Миколаїв
P2	Сидоров С.С.	Київ

2) **Різницею** двох відношень (позначається r – s) є відношення, що утримує усі кортежі, які належать r, але не належать s.

Операція застосовується до відношень з однаковою схемою.

Приклади.

Приклад1.3.

k – m =	(A	B	C)
	a_1	b_1	c_1
	a_2	b_1	c_2

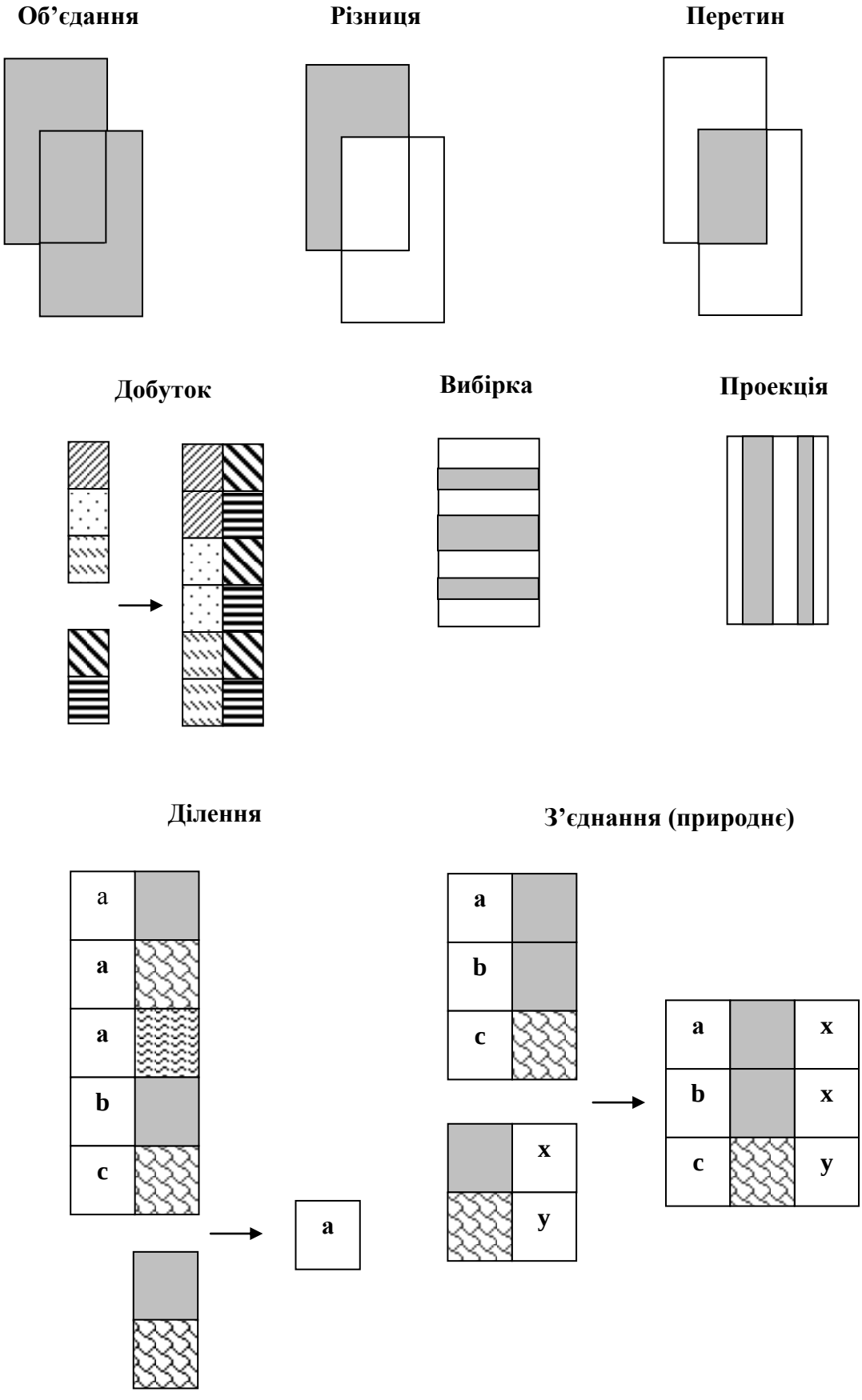


Рис. 1.1. Основні операції реляційної алгебри

Приклад1.4. Різниця відношень R1 та R2 дає постачальників з миколаєва, які не поставляють деталь D1:

R1-R2

№_П	ПІБ_П	Місто_П
P4	Петров П. П.	Миколаїв

3) **Перетином** двох відношень (позначається r ∩ s) є відноше кортежі, які належать одночасно r та s.

Операція застосовується до відношень з однаковою схемою.

Приклади.

Приклад 1.5.

k ∩ m =	(A	B	C)
	a_1	b_2	c_1

Приклад 1.6. Перетином відношень R1 та R2 будуть усі постачальники з Миколаєва, які поставляють деталь D1:

R1 ∩ R2

№_П	ПІБ_П	Місто_П
P1	Іванов І. І.	Миколаїв

4) **Декартовим добутком** відношення r ступня k1 та відношення s ступненя k2 (r × s) є відношення ступня (k1 + k2), яке утримує такі кортежі, перші k1 елементів яких належать r, а останні k2 елементів належать s.

Приклади:

Приклад 1.7. Дано відношення k та m

k	(A	B)
	a_1	b_1
	a_2	b_1

m	(C	D)
	c_1	d_1
	c_2	d_1
	c_2	d_2

Тоді

k × s =	(A	B	C	D)
	a_1	b_1	c_1	d_1
	a_1	b_1	c_2	d_1
	a_1	b_1	c_2	d_2
	a_2	b_1	c_1	d_1
	a_2	b_1	c_2	d_1
	a_2	b_1	c_2	d_2

Приклад 1.8. Нехай відношення R1 – множина постачальників, а S1 – множина деталей:

R1

№_П	**ПІБ_П**	**Місто_П**
P1	Іванов І. І.	Миколаїв
P2	Сидоров С. С.	Київ

S1

№_Д	**Назва**	**Вага**	**Місто_Д**
Д1	гайка	12	Миколаїв
Д3	гвинт	17	Одеса
Д6	шпилька	19	Миколаїв

Тогда декартово произведение данных отношений:

R1 × S1

№_П	**ПІБ_П**	**Місто_П**	**№_Д**	**Назва**	**Вага**	**Місто_Д**
P1	Іванов І. І.	Миколаїв	Д1	гайка	12	Миколаїв
P1	Іванов І. І.	Миколаїв	Д3	гвинт	17	Одесса
P1	Іванов І. І.	Миколаїв	Д6	шпилька	19	Миколаїв
P2	Сидоров С.С.	Київ	Д1	гайка	12	Миколаїв
P2	Сидоров С.С.	Київ	Д3	гвинт	17	Одеса
P2	Сидоров С.С.	Київ	Д6	шпилька	19	Миколаїв

Спеціальні реляційні операції

5) **Вибірка** (застосовується до одного відношення). Результатом її затосування до відношення r є інше відношення, що представляє собою інше відношення, яке є підмножиною кортежів відношення r з певним значенням у виділеному атрибуті.

Нехай r – відношення зі схемою R, A – атрибут в R та a – елемент з домену A. Тоді $\sigma_{A=a}(r)$ – операція вибірки в r кортежів, в яких значення A равно a. В умові вибірки можна застосовувати константи, логічні операції та операції порівняння.

Приклади.

Приклад 1.9.

$\sigma_{Вага<15}(S1)$

№_Д	Назва	Вага	Місто_Д
Д1	гайка	12	Миколаїв

Приклад 1.10. Дано відношення R1

R1

№_П	№_Д	Кількість
P1	Д1	300
P1	Д3	400
P2	Д1	200
P2	Д4	500
P3	Д1	400

Тоді

$\sigma_{N\_Д="Д1"\ AND\ Кількість>300}(R1)$

№_П	№_Д	Кількість
Р3	Д1	400

6) **Проекція** (затососовується до одного відношення) – операція вибору підмножини стовпців. Нехай r – відношення зі схемою R, X – підмножина з R. Проекція r на X $(\pi_X(r))$ є відношення $r'(X)$, яке отримано ви кресленням стовпців, які відповідають атрибутам в (R–X), та виключенням із рядків що залишилися, рядків, які повторюються.

Приклади.

Приклад 1.11. Нехай k – відношення зі схемою ABC:

k	(A	B	C)
	a_1	b_1	c_1
	a_1	b_2	c_1
	a_2	b_1	c_2

Тоді запис $\pi_{C,A}(k)$ означає, що з кожного кортежа, що належить k, формується кортеж довжини 2 з третього та першого його атрибутів у вказаній послідовності:

$\pi_{C,A}(k) =$	(C	A)
	c_1	a_1
	c_2	a_2

← Кортежі, що повторюються, виключаються

Запис $\pi_{C,A}(k)$ еквівалентна запису $\pi_{3,1}(k)$.

Приклад 1.12. Нехай S1 – відношення, що утримує інформацію про деталі.

S1

№_Д	Назва	Вага	Місто_Д
Д1	гайка	12	Миколаїв
Д3	гвинт	17	Одеса
Д6	шпилька	19	Миколаїв

Тоді

$\pi_{1,4}(S1)$

№_Д	Місто_Д
Д1	Миколаїв
Д3	Одеса
Д6	Миколаїв

7) **Ділення**. Нехай r – відношення зі схемою R, s – відношення зі схемою S та $S \subseteq R$. Покладемо $R' = R - S$. Частка від ділення r на s ($r \div s$) – це максимальна підмножина r′ множини $\pi_{R'}(r)$, такої, що декартовий добуток r′ та s містяться в r.

Приклади

Приклад 1.13. Дано відношення k та m

k (A	B	C	D)
a_1	b_1	c_1	d_2
a_2	b_2	c_2	d_1
a_3	b_2	c_3	d_3
a_2	b_2	c_4	d_1
a_1	b_3	c_5	d_4
a_4	b_1	c_6	d_5

m (C	D)
c_2	d_1
c_4	d_1

Тоді

k ÷ m = (A	B)
a_2	b_2

Приклад 1.14. Є відношення отношение

Право

Пілот	Тип_літака
Іванов	707
Іванов	727
Іванов	747
Петров	707
Петров	727
Сидоров	707
Сидоров	727
Сидоров	747
Сидоров	1011
Макаров	727

Потрібно знайти тих пілотів, які мають правокерувати усіма типами літаків з деякої множини q.

q

Тип_літака
707
727
747

Тоді

Право ÷ q

Пілот
Іванов
Сидоров

8) **З’єднання**.

Природнє з’єднання. Нехай r – відношення зі схемою R, s – відношення зі схемою S та R ∪ S = T. Природним з’єданням відношень r та s (r ⊳⊲ s) є відношення q зі схемою T, яке утримує кортежі, кожен з яких є комбінацією кортежа з r та кортежа з s з рівними (R ∩ S) – значеннями.

Примітка. Якщо R ∩ S = ∅, тоді r ⊳⊲ s є декартовим добутком r та s.

Приклади.

Приклад 1.15. Дано відношення k та m

k (A	B)
a_1	b_1
a_1	b_2
a_2	b_1

m (B	C)
b_1	c_2
b_2	c_1

Тоді

k ⊳⊲m = (A	B	C)
a_1	b_1	c_2
a_2	b_1	c_2
a_1	b_2	c_1

Приклад 1.16. Дано відношення R1 та S1

R1

№_П	ПІБ_П	Місто_П
Р1	Іванов І.І.	Миколаїв
Р2	Сидоров С. С.	Київ

S1

№_Д	Назва	Вага	Місто_Д
Д1	гайка	12	Миколаїв
Д3	гвинт	17	Одеса
Д6	шпилька	19	Миколаїв

Природнє з’єднання R1 та S1 за атрибутом *Місто* (у відношенні R1 – це *Місто_П*, а у відношенні S1 – *Місто_Д*):

R1 ⊳⊲ S1

№_П	ПІБ_П	Місто	№_Д	Назва	Вага
Р1	Іванов І.І.	Миколаїв	Д1	гайка	12
Р1	Іванов І.І.	Миколаїв	Д6	шпилька	19

Тета-з’єдання. При природному з’єданні відношення можуть комбінуватися тільки за одноіменними стовпцями та повинні комбінуватися за всіма такими стовпцями.

Примітка. В попере рацію перейменування атрибутів *Місто_Д* и *Місто_П* на *Місто*.

Відношення можуть з'єднуватися також за стовпцями з різними іменами атрибутів, але равними доменами.

Тета-з'єднання відношень r та s за стовпцями i та j $\left(r \underset{i\,\Theta\,j}{\triangleright\!\triangleleft} s \right)$ є множина кортежів у декартовому добутку r та s, таких що i-й елемент r знаходиться у з'язку Θ з j–елементом s.

Якщо Θ є оператором «=», то ця операція зветься **еквіз'єднанням**.

Приклади.

Приклад 1.17. Нехай R1 – відношення, що утримує інформацію про рейси з пункта «A» до пункту «B», S1 – відношення, що утримує інформацію про рейси з пункта «B» до пункту «C».

R1

Номер	Час_вильоту	Час_прибуття
60	9.40	11.45
91	12.50	14.47
112	16.05	18.15
306	20.30	22.25
420	21.15	23.11

S1

Номер	Час_вильоту	Час_прибуття
11	8.30	9.52
60	12.25	13.43
156	16.20	17.40
158	19.10	20.35

Потрібно дізнатися, які рейси з «A» у «B» поєднуються з рейсами з «B» до «C».

Для цього потрібно з'єднати ті кортежі, для *Час_прибуття* у відношенні R1 менша *Час_вильоту* у відношенні S1.

Тета-з'єднання відношень R1 та S1:

Номер	Час_ вильоту	Час_ прибуття	Номер	Час_ вильоту	Час_ прибуття
60	9.40	11.45	60	12.25	13.43
60	9.40	11.45	156	16.20	17.40
60	9.40	11.45	158	19.10	20.35
91	12.50	14.47	156	16.20	17.40
91	12.50	14.47	158	19.10	20.35
112	16.05	18.15	158	19.10	20.35

Приклад 1.18. Дано відношення *Маршрут* и *Адреса*

Маршрут

Номер	Пункт_відправлення	Пункт_призначення
84	Чикаго	Нью-Йорк
109	Нью-Йорк	Лос-Анджелес
117	Атланта	Бостон
213	Нью-Йорк	Бостон
214	Бостон	Нью-Йорк

Адреса

Пілот	**Аеропорт**
Терхьюн	Нью-Йорк
Темпл	Атланта
Тейлор	Атланта
Тарбелл	Бостон
Тодд	Лос-Анджелес
Трумен	Чикаго

← відношення вказує для кожного пілота адресу базового аеропорта, до якого він приписаний

Потрібно призначити пілотів на ті рейси, пункт відправлення яких співпадає з аеропортом, до якого приписаний пілот.

Еквіз'єднання відношень *Маршрут* та *Адреса* за стовпцями *Пункт_відправлення* та *Аеропорт*:

Номер	**Пункт_ відправлення**	**Пункт_ призначення**	**Пілот**	**Аеропорт**
84	Чикаго	Нью-Йорк	Трумен	Чикаго
109	Нью-Йорк	Лос-Анджелес	Терхьюн	Нью-Йорк
117	Атланта	Бостон	Темпл	Атланта
117	Атланта	Бостон	Тейлор	Атланта
213	Нью-Йорк	Бостон	Терхьюн	Нью-Йорк
214	Бостон	Нью-Йорк	Тарбелл	Бостон

Основна відмінність еквіз'єднання від природного в тому, що у останньому стовпці, що з'єднуються, не повторюються.

1.2. Введення в SQL. Категорії команд SQL: DDL, DML, DCL

Після створення бази даних з її об'єктами потрібно працювати, а саме отримувати дані з БД і в тому вигляді, який необхідний в той чи інший момент часу. Для здійснення дій по отриманню даних використовується мова структурованих запитів **SQL**.

SQL (Structured Query Language) – це універсальна мова, яка використовується для створення, модифікації і управління даними в реляційних базах даних. Bahto відміти, що SQL не є мовою програмування. Унікальність даної мови полягає в тому, що вона є єдиною стандартною мовою баз даних. Мову SQL підтримують більше сотні

СУБД. Не вважаєчи на те, що кожна СУБД особливо інтерпретує стандарт (вносить свої доповнення), набір стандартних інструкцій SQL підтримують всі бази даних.

Початок мови SQL був закладений у **1974 році**, коли компанією IBM для експериментальної реляційної СУБД **System R** була розроблена спеціальна мова **SEQUEL (Structured English QUEry Language, структурована англійська мова запитів)**, яка дозволяла дуже просто управляти даними в базі даних. Розробкою займались Дональд Чемберлін і Рей Бойс. Ціллю розробки SEQUEL було створення мови, якою був би взмозі користуватись звичайний користувач, що не має досвіду програмування.

Пізніше мова SEQUEL була переіменована в SQL і в 1986 р. вийшов її перший стандарт ANSI (American National Standards Institute), а в 1987 р. стандарт ISO (International Organization for Standardization, Міжнародна організація по стандартизації). Але вона не була єдиною мовою для баз даних. В той же час Каліфорнійський університет Берклі розробив власну мову **QUEL** для некомерційної реляційної СУБД **Ingres** (прародич сьогоднішньої PostgreSQL). Остання не витримала конкуренцію з SQL.

Таким чином, в 1987 р. SQL стала міжнародною мовою баз даних (SQL-86), в 1992 р. вийшла друга розширена версія даного стандарту (ANSI SQL-92 або SQL2), а в 1999 р. третя версія (SQL:1999, SQL-99 або SQL3). Сьогодні діє стандарт, який був прийнятий в 2003 р. (SQL:2003) з невеликими змінами, які були внесені в 2006 та 2008 роках. Доречі, в новий стандарт SQL була добавлена підтримка роботи і з XML даними.

У випадку, якщо користувачу необхідно отримати дані з бази даних, він звертається з запитом до СУБД на отримання цією інформації. В результаті, СУБД обробляє даний запит і повертає користувачу результатуючі дані. Фактично, **запит** – це команда, написана на мові SQL.

За допомогою SQL можна:

- створювати, модифікувати або видаляти об’єкти бази даних та саму базу даних;
- встановлювати зв’язки між об’єктами бази даних;
- здійснювати вибірку даних;
- вставляти, модифікувати та виділяти дані з бази даних;
- забезпечувати цілісність даних
- інше.

Створення бази даних у СУБД MS Access виконується автоматично при встановленні самої СУБД, тому в даному посібнику не розглядається.

Вищеописані дії та ще багато інших, здійснюються за допомогою інструкцій (команд) SQL, які розглянуто у посібнику.

Всі команди мови SQL поділяють на **чотири категорії**:

1) **DDL** (**D**ata **D**efinition **L**anguage - Мова визначення даних) – включає в себе інструкції, які призначені для створення, модифікації та виділення об’єктів бази даних (таблиці, представлення, індекси тощо).

2) **DML** (**D**ata **M**anipulation **L**anguage - Мова маніпулювання даними) – це набір інструкцій, які призначені для роботи з даними в базі даних, тобто їх вставкою, видаленням, зміною та вибіркою.

3) **DCL** (**D**ata **C**ontrol **L**anguage - Мова управління даними) – містить набір інструкцій, які управляють правами доступу користувачів до об'єктів бази даних.

4) **TCL** (**T**ransaction **C**ontrol **L**anguage – Мова управління транзакціями). В неї вносять набір інструкцій, що дозволяють управляти транзакціями.

За стандартом ANSI підмножина команд DCL є частиною DDL.

У командах SQL не відрізняються прописні та рядкові літери (за виключенням рядкових літералів). Символи та рядки символів заключаються в одинарні лапки, наприклад 'N', 'підручник'. Однорядкові коментарі починають з двох мінусів (--), багаторядкові розташовують між симовлами /* та */.

Кожна команда закінчується символом ';'.

Зауваження. Застосуємо наступні позначення для опису синтаксису:

{ } – зміст дужок розглядається як єдине ціле для інших символів;

| – замінює слово АБО;

[ ] – зміст даних дужок є необов'язковим;

< > – зміст цих дужок замінюється відповідними ключовими словами, літералами, ідентифікаторами або виразами (в залежності від контексту);

… – усе, що передує цим символам, може повторюватися довільну кількість разів;

.,.. – усе, що передує цим символами, може повторюватися довільну кількість разів, кожне входження відділяється комою.

У відповідності зі стандартами ISO ідентиффікатор визначається як послідовність символів довжиною не більш 128, яка починається з літери латинського алфавіту та утримує літери латинського, цифри та знак підкреслення (_). У більшості СУБД накладаються більш жорсткі обмеження на довжину ідентифікатора.

1.3. Типи запитів SQL

Основою роботи з мовою SQL являється запит. **Запит** – це команда, яка повідомляє про необхідність отримання вказаних даних. Наприклад, можна побудувати запит, який буде виводити список усіх клієнтів магазину, дані про найбільш активних покупців або сформувати алфавітний перелік товарів, що користуються найбільшим попитом чи були списані упродовж певного періоду. Як правило, ця інформація отримується з основного джерела реляційної бази даних – таблиць, а результат виводиться на екран комп'ютера. Хоча її можна вивести на принтер, зберегти в файлі або іншому об'єкті бази даних тощо.

Існують наступні основні **типи запитів SQL**:

1. **Запити на вибірку**. Здійснює вибірку даних, що відповідає

вказаним критеріям запиту, з однієї або декількох таблиць. Результат виконання даного запиту – набір записів, що відображаються, як правило, в режимі таблиці.

2. **Запити на зміну.** За допомогою таких запитів можна здійснювати модифікацію даних в базі даних. Існує 4 підтипи запитів на зміну:

- **Запити на поновлення.** Дозволяють поновити дані згідно вказаних умов. Наприклад, встановити нові ціни на товари певного типу, знизивши їх на 10%, у зв'язку з сезонним розпродажем.
- **Запити на додавання.** Дозволяють встановити нові дані в таблицю. Нові дані додаються у кінець вказаної таблиці.
- **Запити на видалення** – це запити, що в результаті своїх дій видаляють записи, які відповідають певному критерію, з вказаної таблиці або групи таблиць. Наприклад, видалити з бази даних товари, дата поставки яких була більше року. Це може бути обумовлено тим, що ці товари попередньо були списані.
- **Запити на створення таблиці.** В результаті роботи даних запитів, здійснюється вибірка даних з однієї або кількох таблиць які заносяться до нової таблиці визначеної структури. Варто відмітити, що такий тип запитів підтримується не усіма СУБД і кожна з них здійснює такий запит різними засобами.

СУБД MS Access підтримує ще два типи запитів:

1. **Перехресні запити** – це запити, результататом роботи яких є організовані в спеціальний формат дані, згруповані за двома критеріями. Як правило, такий тип запитів використовується для формування підсумкових місячних, квартальних або річних звітів по продажам або поставкам товарів.

2. **Запити з параметрами** – це спеціальний інтерактивний тип запиту, який перед виконанням виводить діалогове вікно з проханням ввести один або кілька параметрів, необхідних для його роботи. Ці параметри фактично дозволяють користувачу накладати умови відбору записів. Наприклад, відбирати товари, які були поставлені в інтервалі певних дат, тощо. В інших СУБД цей вид запитів представлений окремими об'єктами бази даних, таких як збережені процедури та функції.

1.4. Підготовка платформи

В MS Access існує два **методи створення запитів**:

1. за допомогою QBE (Query By Example) - на основі шаблону;
2. за допомогою мови структурованих запитів SQL (Structure Query Language).

Розглядаємо тільки другий варіант створення запитів, тобто за допомогою інструкцій мови SQL, оскільки цей спосіб набагато гнучкіший.

Для того, щоб написати запит на мові SQL необхідно обрати об'єкт **«Запросы»** та створити його в довільному режимі. Найзручніший та найшвидший режим створення - за допомогою конструктора (рис. 1.2).

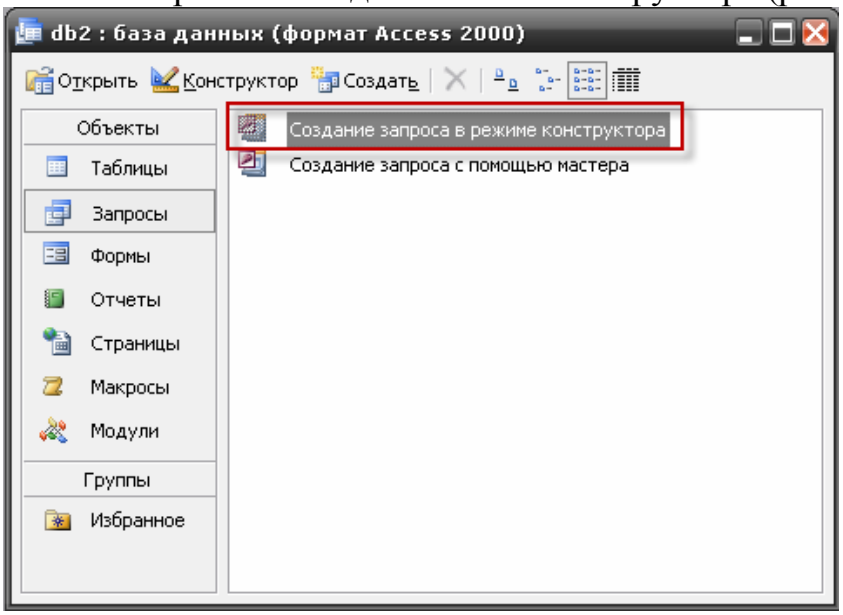


Рис. 1.2. Вікно БД на вкладці «Запити»

Після цього з'явиться вікно додавання таблиць, в якому потрібно обрати таблиці для запиту (рис. 1.3.). На основі цих таблиць буде здійснена вибірка даних з бази даних, тобто з них буде обиратися необхідна інформація. Це потрібно лише у випадку, якщо потрібно створювати запит в режимі QBE. Оскільки наша мета – створення запитів мовою SQL, відмовляємось від запропонованого (кнопка «**Закрыть**»).

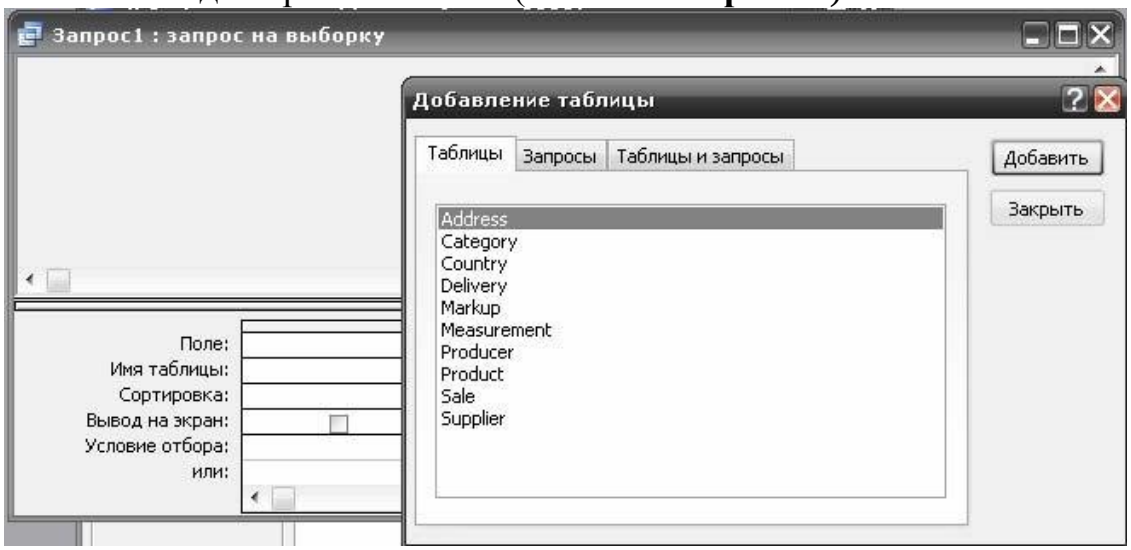


Рис. 1.3. Вікно «Додавання таблиці» у режимі конструктора запитів

Наступний крок – це **перехід в режим SQL**. Для цього на панелі інструментів з випадаючого списку «Вид» виберемо потрібний нам режим (рис. 1.4.).

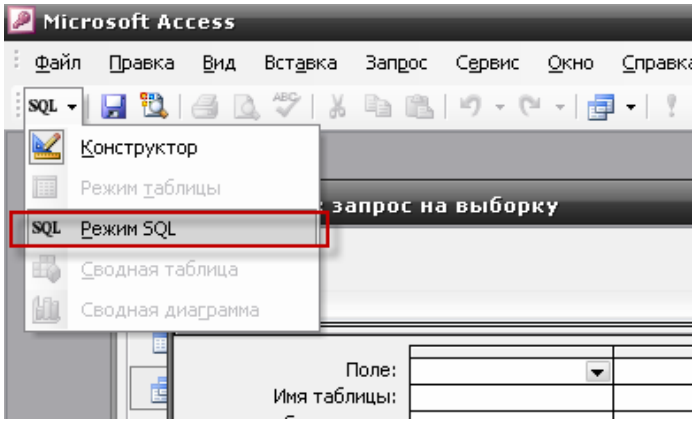


Рис. 1.4. Пункт для переходу до режима SQL

Це дозволить відкрити вікно з редактором для SQL запитів. Коли запит написаний, щоб побачити результат його роботи, тобто запустити його на виконання, достатньо або перейти в режим таблиці, або скористатись кнопкою на панелі інструментів **«Запуск»** (рис. 1.5.).

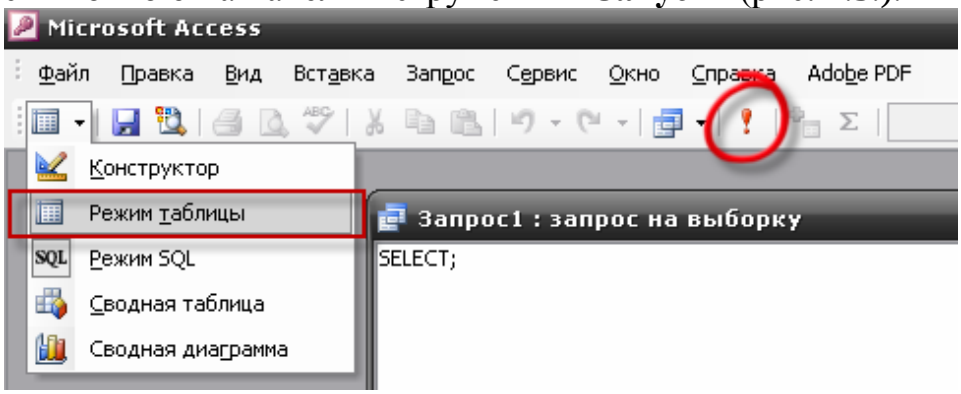


Рис. 1.5. Кнопка «Запуск» для виконання SQL-коду.

1.5. Створення таблиць

Створення нового порожнього відношення (таблиці) виконується за допомогою команди DDL **CREATE TABLE**. Спрощений синтаксис цієї команди :

```
CREATE TABLE <ім'я таблиці>
   (<ім'я поля1> <тип даних> [ (<розмір>) ]
     [NOT] NULL] [ DEFAULT <вираз> ]
     [<обмеження цілісності поля>…]
   [, < ім'я поля2> < тип даних > [ (<розмір >) ]
     [[NOT] NULL] [ DEFAULT < вираз > ]
     [<обмеження цілісності поля >] .,..]
     [, < обмеження цілісності таблиці >] );
```

Пояснення елементів опису наведено у табл. 1.

Таблиця 1.1. Опис елементів команди CREATE TABLE

Елемент	Опис
<ім’я таблиці>	Ім’я таблиці, звичайний ідентифікатор. Повинно бути унікальним в межах бази даних (підсхеми БД)
<ім’я поля>	Ім’я поля (стовпця) таблиці, звичайний ідентифікатор.
<тип даних>	Тип даних поля. Можна використовувати один з наступних типів: – NUMERIC[(довжина [, точність])], NUMBER[(довжина [, точність])] – числа з фіксованою комою; – FLOAT, REAL – дійсні числа; – CHAR[(довжина)], VARCHAR(довжина) – символьні рядки фіксованої або змінної довжини; – DATE, DATETIME – дата/час; – TIME – час (є не в усіх СУБД). Додатково див. Додаток А, таблиця А.1.
<розмір>	Розмір поля в символах (для текста та чисел)
<обмеження цілісності>	– PRIMARY KEY – первинний ключ (обов’язковий, унікальний та єдиний у таблиці); – UNIQUE – унікальність значення поля в межах стовпця таблиці; – CHECK (<умова>) – умова, якій повинна задовольняти значення поля; – REFERENCES <ім’я таблиці> [(<ім’я стовпця>)] – зовнішній ключ.
<обмеження цілісності таблиці>	– CHECK (<умова>) – умова, якій повина задовольняти значення одного або декількох полів; – FOREIGN KEY [(<список полів>)] REFERENCES < ім’я таблиці > [(<список полів>)] – зовнішній ключ; – PRIMARY KEY (<список полів>) – первинний ключ; – UNIQUE (<список полів>) – унікальне значення комбінації полів в межах таблиці.

Якщо розмір поля не вказано, то приймається значення, прийняте в даній СУБД за замовчуванням для вказаного типу. Для усіх СУБД точність для числових типів за замовчуванням дорівнює 0, розмір поля типу CHAR за замовчуванням дорівнює 1, а для типу VARCHAR розмір вказувати необов’язково.

Для типу `DATE` підтримується аріфметика дат, наприклад:

(<поточна дата> + 1) – завтра

(<дата1> – <дата2>) – кількість днів, які пройшли між двома датами;

(<поточна дата> + 1/24) – через годину (для типу дата-час).

Отримати поточну дату можливо за допомогою спеціальної функції, ім’я якої залежить від СУБД: sysdate – для Oracle; now() – для Access и MySQL; getdate() – для MS SQL Server и т.д.

Стандарт SQL включає поняття невизначеного або невідомого значення – NULL-значення. Для обов'язкових полів встановлюється обмеження NOT NULL. Це означає, що при зміні значення цього поля або при додаванні нових записів таблиці значення цього поля не може бути невизначеним (NULL). Обмеження NOT NULL можна накласти на поле тільки єдиний раз, інакше виникає помилка.

Для будь-якого поля за допомогою конструкції DEFAULT <вираз> може бути задано значення за замовчуванням. Воно застосовується в тих випадках, коли при додаванні даних в таблицю значення цього поля не вказується.

Для обмеження цілісності можна задавати імена:

CONSTRAINT <им'я> <обмеження цілісності>

Приклади створення таблиць навчальної бази «Магазин» (структуру БД «Магазин» див. у додатку Б, рис. Б.1.).

Приклад 1.19. Таблиця «Country» з полями IdCountry (первинний ключ), Name:

```
CREATE TABLE Country
(IdCountry VARCHAR(3) NOT NULL CONSTRAINT pk_prod
  PRIMARY KEY,
Name VARCHAR(50) NOT NULL);
```

Приклад 1.20. Таблиця «Address» з полями IdAddress (первинний ключ), IdCountry (зовнішній ключ), Town, Street:

```
CREATE TABLE Address
  (IdAddress INT CONSTRAINT pk_addr PRIMARY KEY,
   IdCountry   VARCHAR(3)   CONSTRAINT   ref_count
    REFERENCES Country,
  Town VARCHAR(20) NOT NULL,
  Street VARCHAR(40));
```

Приклад 1.21. Таблиця «Producer» з полями IdProducer (первинний ключ), Name, IdAddress (зовнішній ключ):

```
CREATE TABLE Producer
(IdProducer NUMERIC(2) CONSTRAINT pk_prod PRIMARY
  KEY,
Name VARCHAR(50) NOT NULL,
IdAddress   INT   NOT   NULL   CONSTRAINT   ref_addr
  REFERENCES Address);
```

Розглянемо ще приклади створення таблиць (без прив'язування до навчальної БД «Магазин»)

Приклад 1.22. Таблиця «Відділи» з полями "Номер відділу" (ПК), "Назва відділа":

```
CREATE TABLE depart
( depno NUMERIC(2) CONSTRAINT pk_dep PRIMARY KEY,
name VARCHAR(80) NOT NULL);
```

Приклад 1.23. Таблиця «Співробітники» з полями «Номер відділа» (зовнішній ключ), «Табельний номер співробітника» (ПК), «ПІБ співробітника», «Посада», «Оклад», «Дата народження», «Телефон», «Дата прийняття на роботу»:

```
CREATE TABLE emp
(depno NUMERIC(2) CONSTRAINT ref_dep REFERENCES
depart,
tabno CHAR(5) CONSTRAINT pk_emp PRIMARY KEY,
name VARCHAR(50) NOT NULL,
post VARCHAR(35) NOT NULL,
salary NUMERIC(7,2) NOT NULL
CONSTRAINT check_salary CHECK (salary > 4600),
born DATE NOT NULL,
phone CHAR(11),
edate DATE DEFAULT trunc(sysdate));
```

Примітка: функція *sysdate* повертає дату та час, тому слід за допомогою функції *trunc* (скорочення відт truncate) встановлювати час в 0 годин, 0 хвилин, 0 секунд.

Приклад 1.24. Таблиця «Діти співробітників» з полями «Табельний номер батька» (зовнішній ключ), «Ім’я дитини», «Стать», «Дата народження»:

```
CREATE TABLE children
(tabno CHAR(5) CONSTRAINT ref_emp REFERENCES
    emp(tabno),
name VARCHAR(50) NOT NULL,
sex CHAR,
born DATE,
CONSTRAINT pk_child PRIMARY KEY (tabno, name),
/* составной первичный ключ */
CONSTRAINT check_sex CHECK (sex IN ('м', 'ж')));
```

Зверніть увагу:

- Загальні обмеження цілісносі та складені ключі вказуються через кому після останнього поля;
- якщо зовнішній ключ посилається на первинний ключ (РК) іншого відношення, імена полів РК можна не вказувати (див. створення таблиці emp);
- типи полів зовнішнього ключа (FK) повинні співпадати з типами полів первинного (або унікального) ключа, на який він посилається;

- якщо зовнішній ключ складений, список полів, що входить до ключа, вказується після перерахування усіх полів таблиці з ключовим словом FOREIGN KEY:

```
create table exam     - «Розклад заліків», основна таблиця
  ( eclass NUMERIC(3),- аудиторія
    edate DATE,       - дата
    edisc VARCHAR(60),- дисципліна
    UNIQUE (eclass, edate));

create table tab      - «Заліки», підпорядкована таблиця
  ( tid NUMERIC(6) PRIMARY KEY,
    tclass NUMERIC(3), - аудиторія
    tdate DATE,        - дата
    tgroup CHAR(6),    - група
    FOREIGN   KEY   (tclass,   tdate)   REFERENCES
exam(eclass, edate));
```

Знайомство з командами SQL розпочнемо з розгляду DML команд, а саме тих, які відповідають за вибірку даних на екран.

1.6. Запити з використанням однієї таблиці. Оператор SELECT

Всі запити на отримання будь-якої кількості даних з однієї або кількох таблиць виконуються за допомогою оператора

SELECT, який має наступний синтаксис:

```
SELECT  [ALL | DISTINCT] { * | поле_для_вибірки [,
        поле_для_вибірки] }
FROM {таблиця | представлення} [as] [псевдонім] [,
     {таблиця | представлення} [as] [псевдонім] ]
[WHERE  умова]
[GROUP BY [ALL] вираз_групування]
[HAVING умова_на_групу]
[ORDER BY імя_поля | номер_поля [ ASC | DESC ] ];
```

Розшифруємо:

- **SELECT** – задає перелік полів, звідки необхідно вибрати дані. Оператор ALL визначений стандартом, але підтримується не усіма СУБД;
- **FROM** – з яких таблиць чи таблиці потрібно здійснити вибірку (де містяться вказані для вибірки поля);
- **WHERE** – умова на вибірку;
- **GROUP BY** – задає перелік полів, по якому необхідно групувати

вибірку, щоб отримати для кожної групи єдине агреговане значення. Для цього в операторі SELECT потрібно ОБОВ'ЯЗКОВО використовувати SQL-функції агрегування;

- **HAVING –** накладає умову, яка дозволяє отримати в результаті лише ті групи, які задовольняють вказаному критерію;
- **ORDER BY** – дозволяє відсортувати результуючу сукупність за вказаним полем.

Крапка з комою являється обов'язковою в кінці кожного SELECT запиту, але деякі СУБД дозволяють її ігнорувати. Також варто відмітити, що допускається вкладеність операторів SELECT. Але робити її необхідно з розумом і лише у випадку необхідності.

Зміст таблиць навчальної БД «Магазин» (файл Shop.mdb) див. у додатоку Б, таблиці Б.1-Б.10.

Приклад 1.25. Створити запит на вибірку всієї інформації про товари:

```
SELECT *
FROM Product;
```

Приклад 1.26. Якщо потрібно отримати лише назву та ціну товару, то слід біля оператора SELECT написати перелік полів для вибірки:

```
SELECT Name, Price
FROM Product;
```

Результат див. рис. 1.6.

Запрос1 : запрос на выборку

Name	Price
Хліб білий	4,5
Вода	2
Хліб чорний	3,5
Ватрушка	4,3
Ковбаса"Рум'янец"	50
Фарш "Добрий"	64,5
Цукерки "Радість"	43
Цукерки"Крабіки"	16
Вода "Лімон"	12,6
Моршинська	4,6
Міргородська	5,2
Фарш"Добриня"	42
Молоко"Корівка"	8,1
Йогурт"Живинка"	5,4

Запись: 1 из 17

Рис. 1.6. Результат запиту з приклада 1.26

При цьому можна одночасно відформатувати вивід інформації на екран, тобто задати назви полів (псевдоніми полів) в результуючому запиті. Для цього запит потрібно *переписати наступним чином:*

Приклад 1.27.

```
SELECT Name as [Назва товару], Price as [Ціна продажу]
FROM Product;
```

Результат див. рис. 1.7.

Запрос1 : запрос на выбор...

Назва товару	Ціна продажу
Хліб білий	4,5
Вода	2
Хліб чорний	3,5
Ватрушка	4,3
Ковбаса"Рум'янец"	50
Фарш "Добрий"	64,5
Цукерки "Радість"	43
Цукерки"Крабіки"	16
Вода "Лімон"	12,6
Моршинська	4,6
Міргородська	5,2
Фарш"Добриня"	42
Молоко"Корівка"	8,1
Йогурт"Живинка"	5,4

Запись: 13 из

Рис. 1.7. Результат запиту з приклада 1.27

Щоб зробити результат запиту ще більш наочнішим, використовують літерали. **Літерал** – це рядок, який записується в подвійних або одинарних лапках та включається в параметр **<поле_для_вибірки>**. Він виводиться в результуючому запиті як окреме поле. В результуючій сукупності даних літерал виступає в ролі тексту, який пояснює значення наступного за ним поля.

Синтаксис додавання літералу наступний:

```
SELECT 'літерал' [, 'літерал']
```

З використанням літералів попередній запит набуде вигляду:

Приклад 1.28.

```
SELECT Name as [Назва товару], Price as [Ціна продажу],
       'грн.' as [Грошова одиниця]
FROM Product;
```

Результат див. рис. 1.8.:

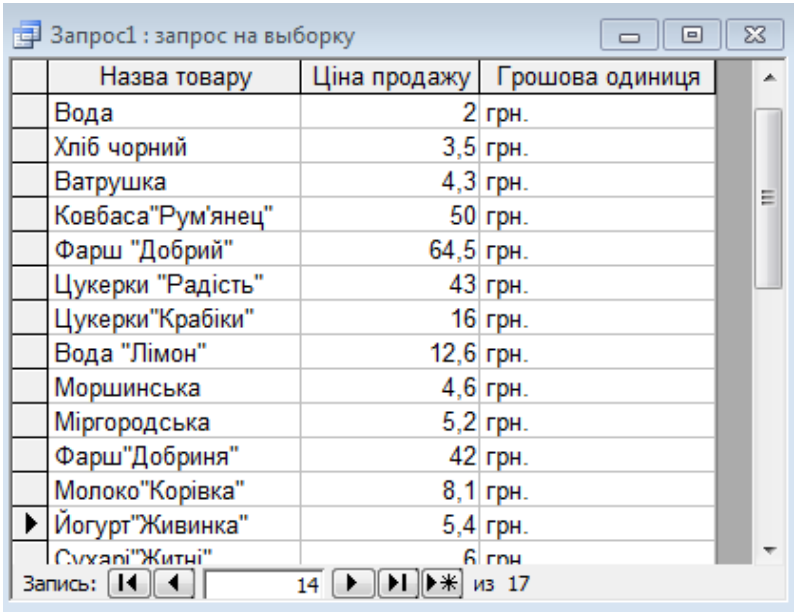


Рис. 1.8. Результат запиту з приклада 1.28

1. Також можна включати до вибірки результати розрахунків.

Приклад 1.29. Додати до вибірки ціну продажу товарів, збільшену в два рази.

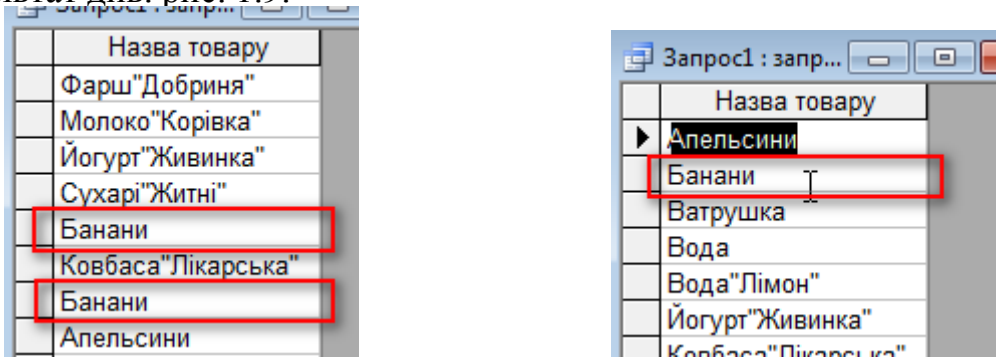
```
SELECT Name as [Назва товару], Price as [Ціна
    продажу], 'грн.' as [Грошова одиниця],
    Price * 2 as [Подвійна ціна продажу]
FROM Product;
```

2. *Для того, щоб уникнути дублювання, потрібно доповнити запит ключовим словом* ***DISTINCT*** *(різний).*

Приклад 1.30.

```
SELECT DISTINCT Name as [Назва товару]
FROM Product;
```

Результат див. рис. 1.9.



а) без DISTINCT б) з DISTINCT

Рис. 1.9. Результат запиту з приклада 1.30

3. До засобів форматування результуючої вибірки можна віднести операцію конкатенації - **&**. За допомогою оператора конкатенації можна вивести об'єднані дані кількох полів в одному.

Приклад 1.30. Вивести інформацію про націнки в одному полі, тобто назва націнки та її розмір:

```
SELECT Markup.Name & ' - ' & Markup.Percent &
          ' %'          AS [Націнка]
FROM Markup;
```

Результат див. рис. 1.10:

	Націнка
▶	Акція - 3 %
	Поточний продаж - 0 %
	Розпродаж - 50 %
	Сезонна знижка - 20 %
*	

Рис. 1.10. Результат запиту з приклада 1.30

Здебільшого, таке форматування використовується для виведення повного імені особи або адреси. Для демонстрації використання оператора конкатенації розгянемо приклад.

Приклад 1.31. Припустимо, *в нашій базі даних існує таблиця, яка зберігає інформацію про працівників магазину і необхідно вивести про них інформацію. Ім'я та прізвище повинно зберігатись в окремих полях, але для кращого візуального сприйняття їх доречно вивести в окремому полі*. Такий запит набуде наступного вигляду:

```
SELECT FName&' '&LName as [Повне імя працівника],
          Address as [Адреса]
FROM Employee;
```

1.7. Вибірка з використанням оператора WHERE

Прості вибірки даних застосовуються рідко. Як правило, відбирається набір даних, які задовольняють певному критерію, умові або їх набору. Для цього в операторі SELECT використовується конструкція **WHERE**, після якої вказується власне сама умова відбору у вигляді виразу. В тілі умови можуть використовуватись:

1. **Оператори**, які дозволяють виконувати набір дій над одним або декількома компонентами виразу.
   - **Арифметичні**: додавання (+), віднімання (-), множення (*), ділення (/), цілочисельного ділення одного операнда на

другий (\ ), оператор ділення за модулем (Mod), піднесення в ступень (^). В ролі операндів можуть бути як числа, так і значення полів.

- **Порівняння**: >, <, >=, <=, =, <>. Якщо один з операндів має значення NULL, то результатом порівняння також буде NULL.
- **Логічні (булівські),** результатом роботи яких є логічне значення True (-1), False (0) або NULL. Їх використовують для комбінування результатів виконання двох і більше операцій. Це: логічне І (And), включаюче АБО (Or), логічне заперечення (Not) та виключаюче АБО (Xor).
- **Конкатенації** - для об'єднання кількох рядків (&). Про нього ми вже говорили вище.
- **Оператори SQL**: BETWEEN..AND.., IN, LIKE, IS NULL (перевірка на рівність нулю).

2. **Літерали** – це значення в явному їх представленні.

- **Числові**, які можуть містити знак розділення (в десяткових числах) та знак мінус (-) для від'ємних значень, символи e або E. Наприклад: 3,4567E-01, 12000, -25.
- **Текстові (рядкові)** – будь-які друковані символи (А-Я, 0-9, знаки пунктуації тощо). Їх слід писати в одинарних або подвійних лапках. Наприклад: "Мелодрама", "Київ", 'Готель "Україна" '.
- **Дати та часу**. В більшості СУБД дата та час пишуться в одинарних лапках, але це може бути і інший довільний символ. В MS Access цим символом є символ хеш (#). Але ця вимога лише для виразів SQL і при введенні значення дати або часу через користувацький інтерфейс даний символ не вказується. Наприклад: #01.03.99#, #15-янв-2001#.

3. **Функції** дозволяють спростити процес встановлення умови для вибірки даних. В MS Access можна використовувати як вбудовані функції, так і визначені користувачем (написані на VBA). Наведемо короткий перелік вбудованих функцій:

- *Дати та часу*:
  - Date() – повертає поточну дату;
  - DateAdd("d", -15, [ДатаПоставки]) - повертає дату, що на 15 днів передує даті, заданої значенням поля «ДатаПоставки»;
  - DateDiff("d", [ДатаПоставки],[ДатаПродажу]) – повертає значення, що є різницею значень полів «ДатаПоставки» та «ДатаПродажу»;
  - Year(#12.06.14#) - повертає рік з вказаної дати, тобто 2014
- *Рядкові*:
  - Format(Date, #dd-mm-yyyy#) – повертає відформатовану дату;

- InStr(“Місто”,”С”) – повертає число, що вказує позицію першого входження одного рядка в інший, тобто 3;
- LCase (“МІСТО”) – переводить рядок в нижній регістр;
- Left([Місто],2) – відображає два перших сивола значення поля «Місто»;
- Right([Місто],3) - відображає три останніх сивола значення поля «Місто»;
- Trim([Назва]) – повертає значення поля «Назва» без пропусків (space).

•*Приведення типу даних:*

- Val(“12.35”) – конвертує текст в число, тобто повертає 12,35;
- Str(12,35) – конвертує число в текст, – “12,35”.

Додатково див. Додаток А, таблиці А.2-А.6.

Наведемо кілька прикладів створення вибірки з умовами:

Приклад 1.32. Запит на вибірку інформації про товар з назвою «Банани», в якому крім назви товару потрібно вказати його категорію, виробника та ціну:

```
SELECT Name, IdCategory, IdProducer, Price
FROM Product
WHERE Name = 'Банани';
```

Результат див. на рис. 1.11.

Запрос1 : запрос на выборку

	Name	IdCategory	IdProducer	Price
▶	Банани	7	12	15,7
	Банани	7	6	10
*		0	0	0

Запись: 1 из 2

Рис. 1.11. Результат запиту з приклада 1.32

Слід відмітити, що насправді замість категорії і виробника СУБД виводить їх відповідний ідентифікатор, але під час роботи з MS Access ці дані приховуються. Для виведення значення за ідентифікатором використовуються багатотабличні запити, які будуть розглянуті пізніше (п. Багатотабличні запити на вибірку. Декартова множина значень).

Приклад 1.33.Створити запит на вибірку інформації про товари, ціни яких знаходяться в межах від 10 до 50 грн.

```
SELECT Name, IdCategory, IdProducer, Price
FROM Product
WHERE Price >= 10 AND Price <= 50;
```

Результат див. на рис. 1.12

Запрос1 : запрос на выборку

Name	IdCategory	IdProducer	Price
Ковбаса"Рум'янец"	4	5	50
Цукерки "Радість"	5	6	43
Цукерки"Крабіки"	5	10	16
Вода "Лімон"	2	8	12,6
Фарш"Добриня"	4	5	42
Банани	7	12	15,7
Банани	7	6	10

Рис. 1.12. Результат запиту з приклада 1.33

Є і простіший спосіб написання такого роду запиту, коли критерієм відбору слугує діапазон значень. Полягає він у використанні оператора SQL **BETWEEN..AND...** *З використанням даного оператора запит набуде вигляду з* приклада 1.34.

Приклад 1.34.

```
SELECT Name, IdCategory, IdProducer, Price
FROM Product
WHERE Price BETWEEN 10 AND 50;
```

Результат буде аналогічний, за виключенням того, що остання відібрана ціна буде 49, тобто 50 не включається.

У випадку, якщо необхідно здійснити повністю протилежну операцію, тобто *вивести товари, ціни яких не входять у вказаний проміжок*, слід використати оператор логічного заперечення NOT: див. приклад 1.35.

Приклад 1.35.

```
SELECT Name, IdCategory, IdProducer, Price
FROM Product
WHERE Price NOT BETWEEN 10 AND 50;
```

Оператор **BETWEEN..AND..** можна використовувати для перевірки потрапляння в проміжок дат, літер тощо.

Примітка! Дата записується в форматі #00/00/0000#. Для отримання поточної дати використовується функція Date().

Приклад 1.35. Створити запит на вибірку інформації про поставку товарів в інтервалі від 01/06/2014 до поточної дати.

```
SELECT IdProduct, DateDelivery
FROM Delivery
WHERE DateDelivery BETWEEN #01/06/2014# AND Date();
```

Результат див. на рис. 1.13:

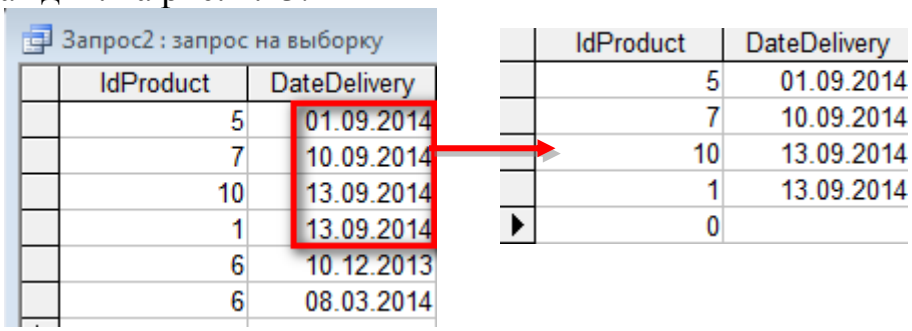


Рис. 1.13. Результат запиту з приклада 1.35

Приклад 1.36. Створити запит на вибірку інформації про товари, ціна яких 10 і 50 грн.

```
SELECT Name, IdCategory, IdProducer, Price
FROM Product
WHERE Price=10 OR Price=50;
```

Результат див. на рис. 1.14.

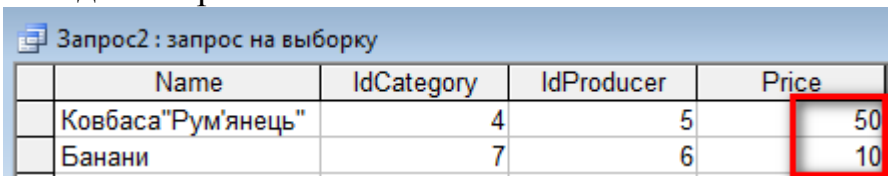


Рис. 1.14. Результат запиту з приклада 1.36

Даний запит теж можна скоротити, використавши оператор **IN**, який дозволяє перевірити належність даних поля множині визначених значень.

Приклад 1.37.

```
SELECT Name, IdCategory, IdProducer, Price
FROM Product
WHERE Price IN(10, 50);
```

Для пошуку значень за шаблоном використовується оператор **LIKE**. Шаблон виразу може включати:

- * або % - в даній позиції може бути присутній 0 або більше символів;
- ? або _ – в даній позиції обов'язково присутній один довільний символ;
- # - в даній позицій присутня одна цифра;
- [a-z] - в даній позиції обов'язково присутній один символ з вказаного діапазону;
- [abc] - в даній позиції обов'язково присутній один символ з вказаного діапазону значень;

- [!a-z] - в даній позиції обов'язково присутній один символ, що не входить у вказаний діапазон;
- [!abc] - в даній позиції обов'язково присутній один символ, що не входить у вказаний діапазон значень.

Приклад 1.38. Створити запит на вибірку інформації про товари, в назві яких не меньше двух літер "о".

```
SELECT Name, IdCategory, IdProducer, Price
FROM Product
WHERE Name LIKE '*o*o*';
```

Результат див. на рис. 1.15.

Запрос2 : запрос на выборку

Name	IdCategory	IdProducer	Price
Вода "Лімон"	2	8	12,6
Міргородська	6	11	5,2
Молоко"Корівка"	3	10	8,1

Рис. 1.15. Результат запиту з приклада 1.38

Приклад 1.39. Використовуючи маску, створити *запит на вибірку інформації про товари, назви яких починаються з літер, які лежать в проміжку від А до К:*

```
SELECT Name, IdCategory, IdProducer, Price
FROM Product
WHERE Name BETWEEN "A%" and "K%";
```

Результат див. на рис. 1.16.

Запрос2 : запрос на выборку

Name	IdCategory	IdProducer	Price
Вода	2	22	2
Ватрушка	1	4	4,3
Вода "Лімон"	2	8	12,6
Йогурт"Живинка"	3	10	5,4
Банани	7	12	15,7
Банани	7	6	10

Рис. 1.16. Результат запиту з приклада 1.39

Але, результуюча вибірка не буде виводити на екран товари, які починаються з літери «К», оскільки оператор BETWEEN..AND.. не включає кінцеве вказане значення і про це ми вже сьогодні говорили.

Приклад 1.40. Якщо необхідно *вивести товари, які починають з літери А по К включно, тоді умову слід перебудувати:*

```
SELECT Name, IdCategory, IdProducer, Price
FROM Product
WHERE Name BETWEEN "A%" and "Л%";
```

В SQL існує поняття **NULL поля**. Значення NULL не рівносильно пропуску (space) або нулю в даних числового типу. Поле даних має значення NULL, якщо йому не було надане значення, тобто воно порожнє. Фактично NULL є недійсним значенням. Знайти записи, які містять NULL-значення можна за допомогою ключових слів **IS NULL** або **IS NOT NULL** (в залежності від суті вибірки).

Приклад 1.41. Створити запит на вибірку інформації про товари, які мають ціну, тобто значення ціни яких не рівне NULL:

```
SELECT Name, IdCategory, IdProducer, Price
FROM Product
WHERE Price IS NOT NULL;
```

Результатом будуть записи про товари, для яких встановлена ціна.

Дуже часто виникає необхідність вказати кілька критеріїв відбору. В такому випадку їх можна поєднати за допомогою логічного оператора **AND** або **OR** (вибір залежить від необхідного результату).

Приклад 1.42. Написати запит на вибірку даних про товари, ціна яких більше 40 грн. і наявна кількість знаходиться в діапазоні від 50 до 100.

```
SELECT Name, Price, Quantity
FROM Product
WHERE Price >=40 AND Quantity BETWEEN 50 AND 100;
```

Результат запиту див. на рис. 1.17

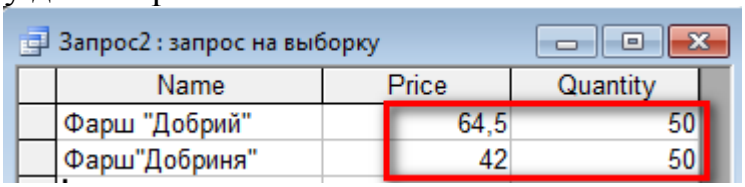


Рис. 1.17. Результат запиту з приклада 1.42.

Якщо замість оператора AND вказати оператор OR, тоді на екран будуть виведені дані, які задовільняють або одну, або другу умову, або ж їх поєднання (рис. 1.18.).

Name	Price	Quantity
Хліб білий	4,5	100
Ковбаса"Рум'янець"	50	20
Фарш "Добрий"	64,5	50
Цукерки "Радість"	43	32
Фарш"Добриня"	42	50
Молоко"Корівка"	8,1	75
Ковбаса "Лікарська"	63,8	15

Рис. 1.18. Результат запиту з приклада 1.42 при зміні оператора AND на OR

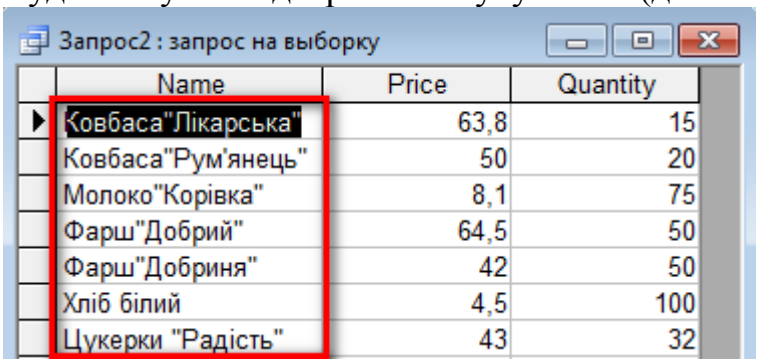
1.8. Сортування даних

Для сортування результуючого набору в операторі SELECT використовується ключове слово **ORDER BY** з необов'язковим параметром **ASC** (за замовчуванням) – сортування за зростанням, або **DESC** – сортування за спаданням.

Приклад 1.43. Відобразити інформацію про товари, відсортовані за зростанням їх назв:

```
SELECT Name, Price, Quantity
FROM Product
WHERE Price >=40 OR Quantity BETWEEN 50 AND 100
ORDER BY Name ASC;
```

Результатом буде наступна відсортована сукупність (див. Рис. 1.19):



Name	Price	Quantity
Ковбаса"Лікарська"	63,8	15
Ковбаса"Рум'янець"	50	20
Молоко"Корівка"	8,1	75
Фарш"Добрий"	64,5	50
Фарш"Добриня"	42	50
Хліб білий	4,5	100
Цукерки "Радість"	43	32

Рис. 1.19. Результат запиту з приклада 1.43

Приклад 1.44. Крім імен полів в списку при ORDER BY можна використовувати їх порядкові номери в списку SELECT.

```
SELECT Name, Price, Quantity
FROM Product
WHERE Price >=40 AND Quantity BETWEEN 50 AND 100
ORDER BY 1;
```

Результат буде аналогічний попередньому (рис. 1.43).

Приклад 1.45. *Сортувати можна і за кількома полями.*

```
SELECT Name, Price, Quantity
FROM Product
WHERE Price <= 20 OR Quantity BETWEEN 50 AND 100
ORDER BY 1, 2 DESC;
```

В такому випадку вибірка буде відсортована за назвою продукту за зростанням, а потім за ціною за спадання (див. рис. 1.20).

Запрос2 : запрос на выборку

Name	Price	Quantity
Банани	15,7	30
Банани	10	10
Ватрушка	4,3	20
Вода	2	2
Вода"Лімон"	12,6	26
Йогурт"Живинка"	5,4	10
Міргородська	5,2	5
Молоко"Корівка"	8,1	75
Моршинська	4,6	5
Сухарі"Житні"	6	15
Фарш"Добрий"	64,5	50
Фарш"Добриня"	42	50

Рис. 1.20. Результат запиту з приклада 1.45

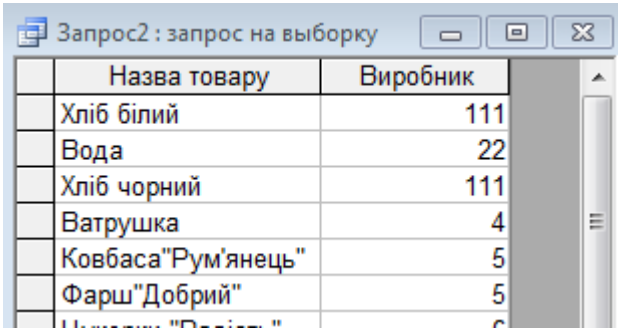
1.9. Багатотабличні запити на вибірку даних. Декартова множина значень

В попередніх розділах ми здійснювали вибірку даних про товари і отримували інформацію лише з однієї таблиці. Якщо нам необхідно було вивести інформацію про виробника або постачальника продукції, то ми вказували поле зовнішнього ключа і дані були виведені на екран.

Приклад 1.46. Побудуємо ще раз такий запит: *виведемо інформацію про товар та його виробника.*

```
SELECT Name as [Назва товару], IdProducer as [Виробник]
FROM Product;
```

Результат наведено на рис. 1.21.



Назва товару	Виробник
Хліб білий	111
Вода	22
Хліб чорний	111
Ватрушка	4
Ковбаса"Рум'янець"	5
Фарш"Добрий"	5

Рис. 1.21. Результат запиту з приклада 1.46

На перший погляд жодних проблем, але насправді замість виробника СУБД виводить їх відповідний ідентифікатор, тобто значення зовнішніх ключів. MS Access ці дані приховує, але при роботі з іншими СУБД результат Вас не задовольнить. Крім того, якщо необхідно буде накласти умову на вибірку за виробником, то можуть виникнути труднощі.

Приклад 1.47. Вивести інформацію про товари виробника ВАТ «Росинка»:

```
SELECT Name as [Назва товару], IdProducer as [Виробник]
FROM Product
WHERE IdProducer= 'ВАТ "Росинка"';
```

Після запуску на виконання такого запиту, згенерується помилка про невідповідність типів, оскільки поле «**IdProducer**» є числовим, а умова символьного типу.

Рішень такої ситуації дві:

1. *Приклад 1.48. Отримати інформацію про те, яке значення первинного ключа відповідає виробнику ВАТ «Росинка»* і переписати запит:

```
SELECT Name as [Назва товару], IdProducer as [Виробник]
FROM Product
WHERE IdProducer= 34;
```

2. *Приклад 1.49. Перебудувати запит і зробити його багатотабличним.*

Звичайно, що перший спосіб не дуже зручний, адже ми не можемо завжди шукати відповідні первинні ключі в зв'язаних таблицях і підставляти їх значення. Отже, розглянемо другий спосіб більш детальніше.

Для того, щоб створити багатотабличний запит, необхідно в списку таблиць при конструкції FROM перелічити список необхідних таблиць. Після цього в списку SELECT (або в місцях де йде звернення до полів таблиць) необхідно вказати, які саме поля і з яких таблиць необхідно отримати.

Доступ до полів таблиць організовується за допомогою операторів ідентифікації крапка (**.**) та знак оклику (**!**) наступним чином:

Класс_Об'єкта!Ім'я_Об'єкта.Властивість_або_метод

Наприклад, якщо необхідно звернутися до поля **Name** таблиці **Student**, то потрібно написати наступний вираз:

```
Student.Name
```

або

```
Tables!Student.Name
```

Приклад 1.50. Створити запит на вибірку даних про товари, з вказанням їх категорії та виробника:

```
SELECT Product.Name as [Назва товару],
       Category.Name as [Категорія],
        Producer.Name as [Виробник]
FROM Product, Category, Producer;
```

Результат наведено на рис. 1.21

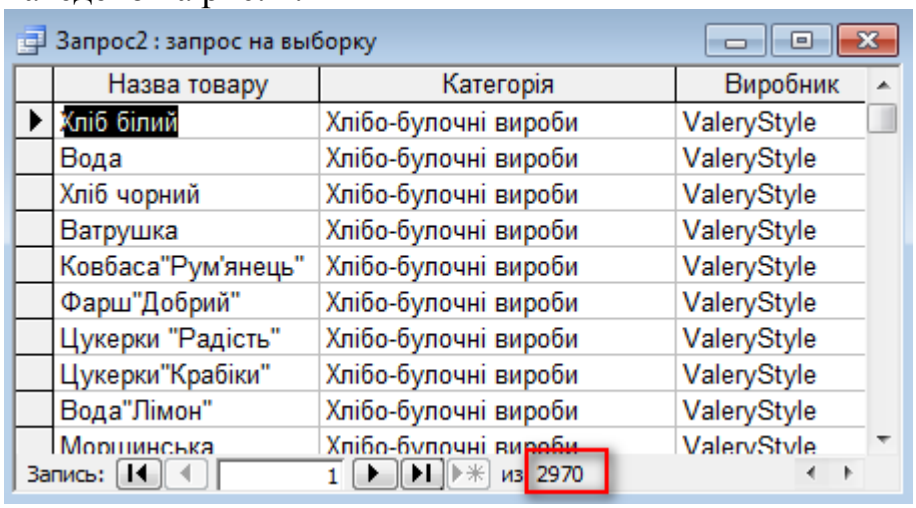


Рис. 1.21. Результат запиту з прикладу 1.50

В результаті роботи запиту утворилась кількість записів, більша за існуючу. В такому випадку говорять, що утворилась **«декартова множина значень»**, тобто всі можливі комбінації записів. Щоб такого не було, потрібно прописати використовувані таблиці зв'язані між собою в тілі оператора WHERE.

Приклад 1.51.

```
SELECT Product.Name as [Назва товару],
       Category.Name as [Категорія],
       Producer.Name as [Виробник]
FROM Product, Category, Producer
WHERE Product.IdCategory=Category.IdCategory
      AND Product.IdProducer=Producer.IdProducer;
```

Результат подано на рис. 1.22.

Запрос2 : запрос на выборку

Назва товару	Категорія	Виробник
Хліб білий	Хлібо-булочні вироби	Хлібзавод№4
Вода	Лікери	Біола
Хліб чорний	М'ясні вироби	Хлібзавод№4
Ватрушка	Хлібо-булочні вироби	ВАТ"Миколаїв-хліб"
Ковбаса"Рум'янець"	М'ясні вироби	ПП"Ряба курка"
Фарш"Добрий"	М'ясні вироби	ПП"Ряба курка"
Цукерки"Радість"	Кондитерські вироби	ЗАТ"Яблуко"
Цукерки"Крабіки"	Кондитерські вироби	ВАТ"Рум'янец"
Вода"Лімон"	Лікери	ВАТ"Карась"
Моршинська	Бакалея	ВАТ"Росинка"
Міргородська	Бакалея	ВАТ"Росинка"

Запись: 7 из 18

Рис. 1.22. Результат запиту з прикладу 1.51

Приклад 1.52. Доповнити запит з приклад 1.51. умовою: відобразити лише ті товари, категорія яких «Фрукти»:

```
SELECT Product.Name as [Назва товару], Category.Name
     as [Категорія], Producer.Name as [Виробник]
FROM Product, Category, Producer
WHERE Product.IdCategory=Category.IdCategory
      AND Product.IdProducer=Producer.IdProducer
     AND Category.Name='Фрукти';
```

Результат подано на рис. 1.23.

Запрос2 : запрос на выборку

Назва товару	Категорія	Виробник
Банани	Фрукти	Banana Republica
Банани	Фрукти	ЗАТ"Яблуко"
Апельсини	Фрукти	ЗАТ"Яблуко"

Запись: 1 из 3

Рис. 1.23. Результат запиту з прикладу 1.52

Приклад 1.53. Відсортувати дану вибірку за назвмих виробників за зростанням:

```
SELECT Product.Name as [Назва товару],
       Category.Name as [Категорія],
       Producer.Name as [Виробник]
FROM Product, Category, Producer
WHERE Product.IdCategory=Category.IdCategory
     AND Product.IdProducer=Producer.IdProducer
     AND Category.Name='Фрукти'
ORDER BY 3;
```

Хоча назви таблиць перед іменами полів достатньо добре інформують користувача про процес вибірки, постійно переписувати їх (інколи достатньо громіздкі назви) доволі важко. Для цього в SQL існує поняття **аліасів (Alias) або псевдонимів** імен таблиць. Псевдоніми використовуються в основному для візуального спрощення тексту запитів.

Приклад 1.54. Переписати попередній запит з використанням псевдонімів таблиць:

```
SELECT pr.Name as [Назва товару], c.Name as [Категорія],
       Producer.Name as [Виробник]
FROM Product as pr, Category as c, Producer
WHERE pr.IdCategory=c.IdCategory AND
    pr.IdProducer=Producer.IdProducer AND
    c.Name='Фрукти'
ORDER BY 3;
```

Також при створенні псевдоніму припустимо упускати ключове слово AS. *Приклад 1.55.*

```
SELECT pr.Name as [Назва товару],
       c.Name as [Категорія],
     Producer.Name as [Виробник]
FROM Product pr, Category c, Producer
WHERE pr.IdCategory=c.IdCategory AND
    pr.IdProducer=Producer.IdProducer AND
    c.Name='Фрукти'
ORDER BY 3;
```

1.10. Функції агрегування. Необхідність використання

В SQL, крім звичайних функцій, існує ряд спеціальних функцій, які називають **функціями агрегування** або **групування**. Кожна з цих функцій працює з сукупністю значень поля деякої таблиці, а результатом функцій є єдине значення. Саме від принципу їх роботи і походить назва цих функцій. Прикладів необхідності їх використання можна навести

безліч. Наприклад, необхідно вивести суму продажу товарів на кожну дату, загальну кількість проданих будинків, розміщених в окремих районах або ж необхідно визначити мінімальну і максимальну ціни на товари окремої категорії.

Функції агрегації (агрегування) переважно повертають числові результати і зазвичай використовуються для розрахунку певної статистики. В кожній СУБД підтримується свій набір таких функцій, але ряд з них затверджені стандартом SQL та спільні для всіх. Отже, до **функцій агрегування належать:**

- **COUNT** – кількість значень (записів) в полі;
- **SUM** - сума значень поля;
- **AVG** – середнє арифметичне значень поля;
- **MAX** – максимальне значення з сукупності;
- **MIN** – мінімальне значення в сукупності;
- ***StDev** – середньоквадратичного відхилення значень поля;
- ***Var** – дисперсія роз граничних значень, які містяться у вказаному полі;
- ***First** – перший запис з результуючого набору за вказаним полем;
- ***Last** – останній запис з результуючого набору за вказаним полем.

* - підтримуються лише Microsoft Access.

Для функцій SUM і AVG поле *обов'язково* повинно містити числові значення, в інших випадках поле можу містити дані довільного типу.

Приклад 1.56. Створити запит на отримання інформації про загальну кількість товарів, їх загальну та середню вартість.

```
SELECT COUNT(Product.IdProduct) as [Кількість товару],
      SUM(Sale.Price*Sale.Quantity) as [Загальна
      вартість продаж], AVG(Sale.Price*Sale.Quantity)
      AS [Середня вартість продаж]
FROM Product, Sale
WHERE Product.IdProduct=Sale.IdProduct;
```

Результат наведено на рис. 1.24.

Запрос2 : запрос на выборку

Кількість товару	Загальна вартість продаж	Середня вартість продаж
20	583,5	29,175

Запись: 1 из 1

Рис. 1.24. Результат запиту з прикладу 1.56

Приклад 1.57. Розглянемо особливості запиту. *Написати запит, що дозволить підрахувати лише загальну кількість записів в таблиці, що містить інформацію про товари:*

```
SELECT COUNT(*) as КількістьЗаписів
FROM Product;
```

Результат виконання запиту подано на рис. 1.25.

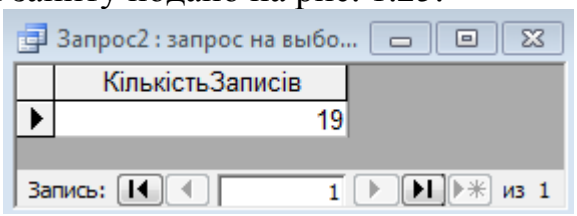


Рис. 1.25. Результат запиту з прикладу 1.57

Функція COUNT з зірочкою в якості параметра включає при підрахунку як NULL-значення, так і значення, що повторюються. Припустимо, виникла ситуація, коли необхідно вивести список товарів з унікальними назвами (цілком імовірна ситуація, коли товар з назвою «Банани» поставляється кількома постачальниками і виробляється кількома виробниками). Для здійснення таких підрахунків згідно стандарту SQL в тілі функцій агрегування, зокрема і COUNT можна використовувати ключове слово **DISTINCT** (різний) з метою підрахунку лише унікальних ненулевих значень, а також ключове слово **ALL** (всі) для врахування при підрахунку всіх значень поля, крім NULL. ALL неявно підставляється по замовчуванню. В такому випадку запит з прикладу 1.57 перепишеться:

Приклад 1.58.

```
SELECT COUNT(DISTINCT Name) as [Кількість
       унікальних назв товарів]
FROM Product;
```

СУБД MS Access не в повній мірі підтримує стандарт ANSI SQL, і тому даний запит викличе помилку. Явне використання ключового слова ALL в будь-якій функції агрегування також в MS Access викличе помилку.

Приклад 1.59. Вивести на екран мінімальну та максимальну ціни на товари категорії «Кондитерські»:

```
SELECT MIN(Product.Price) as [Мінімальна ціна],
     MAX(Product.Price) as [Максимальна ціна]
FROM Product, Category
WHERE Product.IdCategory=Category.IdCategory
AND
      Category.Name = 'Кондитерські вироби';
```

Результат подано на рис. 1.26

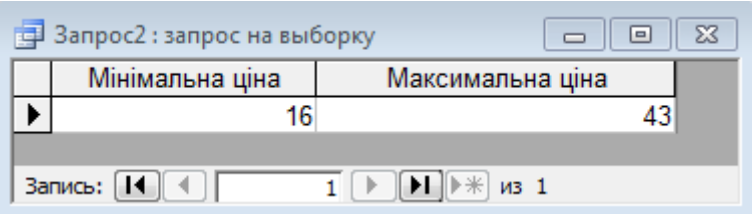


Рис. 1.26. Результат запиту з прикладу 1.59

1.11. Групування даних. Оператори GROUP BY та HAVING

Робота з функціями агрегування відрізняється від звичайної вибірки, оскільки їх результатом є одне єдине значення. У зв'язку з цим при створенні запиту на вибірку неможна одночасно при операторі SELECT використовувати звичайні поля та функції агрегування.

Для реалізації такої можливості використовується поняття групування даних та оператор, що його здійснює - **GROUP BY**. Вказана конструкція дозволяє визначати групи, які володіють спільними характеристиками, а потім генерувати статистику для кожної групи (з використанням нобхідної функції агрегування). В результаті СУБД буде повертати стільки рядків, скільки груп визначається.

Приклад 1.60. Вивести назви товарів та найменші ціни їх продажу:

```
SELECT Product.Name, MIN(Sale.Price) &' грн.'
       as [Мінімальна ціна]
FROM Product, Sale
WHERE Product.IdProduct=Sale.IdProduct
GROUP BY Product.Name;
```

Результат наведено на рис. 1.26.

Поле «Name» з таблиці «Product» в списку SELECT є полем групування. СУБД групує записи за значенням в даному полі, а потім шукає мінімальне значення ціни для кожної окремої групи, і після цього виводиться результат.

При роботі з оператором GROUP BY слід пам'ятати про те, що **всі поля групування, які ви задаєте в списку SELECT, повинні бути включені і в список GROUP BY**. Якщо полів кілька, то їх назви розділяються комами.

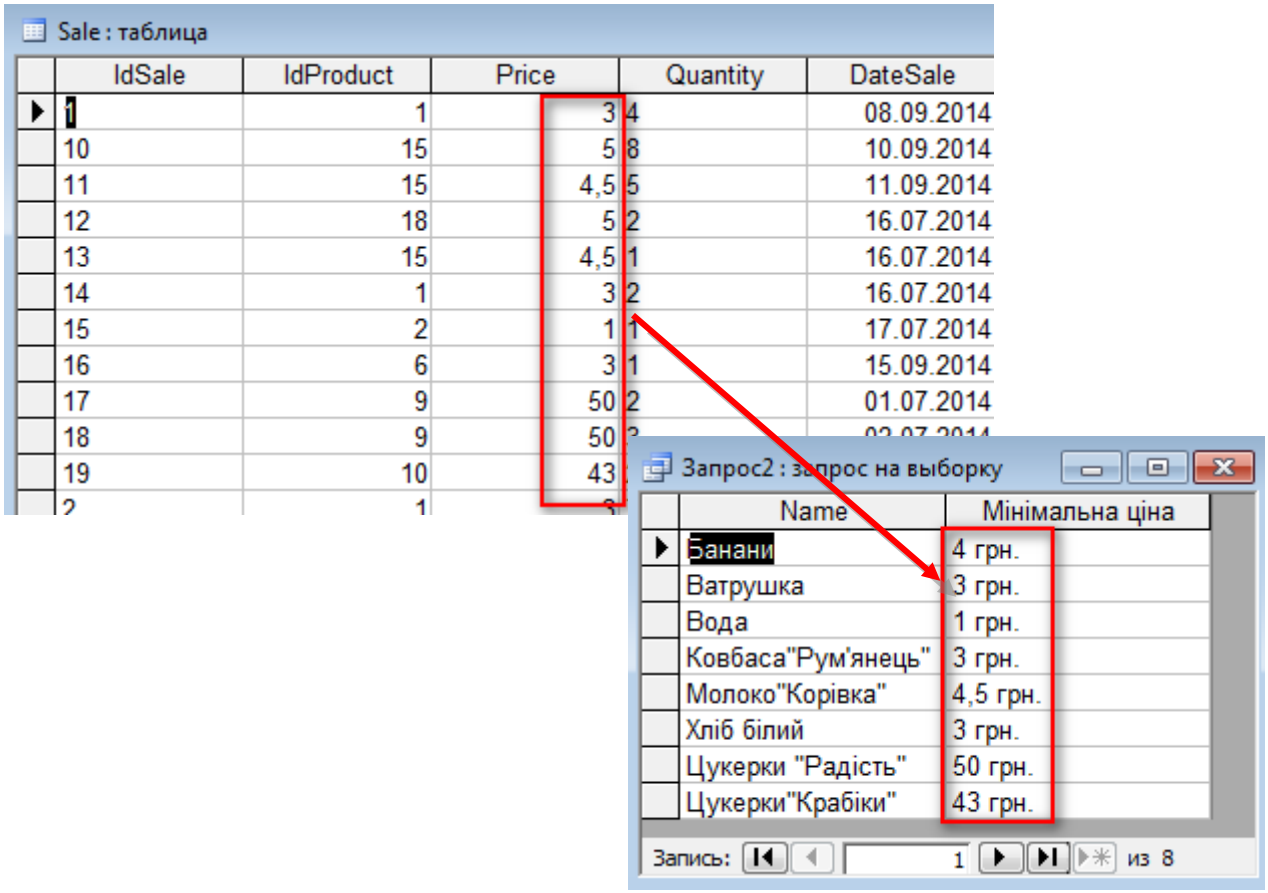


Рис. 1.26. Результат запиту з прикладу 1.60

Приклад 1.61. Вивести на екран категорії продуктів, виробників товарів цих категорій та значення середньої ціни в розрізі цих груп (категорій та виробників). Відсортувати вибірку за назвою виробників.

```
SELECT Producer.Name as [Виробник], Category.Name
     as [Категорія], AVG(Product.Price) &' грн.'
     as [Середня ціна]
FROM Product, Category, Producer
WHERE Product.IdCategory=Category.IdCategory
      AND Product.IdProducer=Producer.IdProducer
GROUP BY Producer.Name, Category.Name
ORDER BY 1;
```

Результат наведено на рис. 1.27

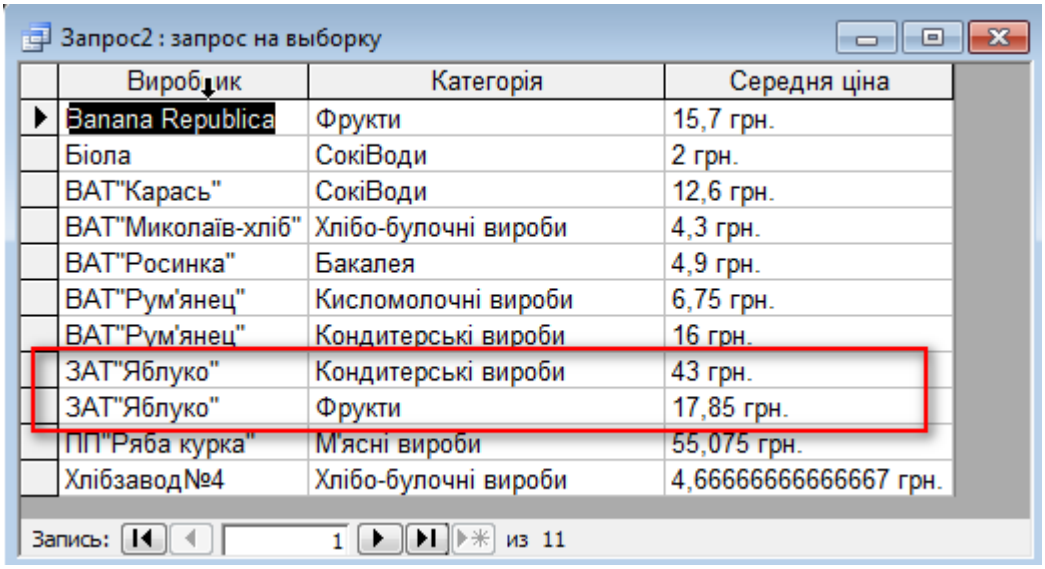


Рис. 1.27. Результат запиту з прикладу 1.61

Приклад 1.62. Вивести інформацію про товари, мінімальна ціна продажу яких була більше 10 грн.:

```
SELECT Product.Name, MIN(Sale.Price)&' грн.'
        as [Мінімальна ціна]
FROM Product, Sale
WHERE Product.IdProduct=Sale.IdProduct
GROUP BY Product.Name
HAVING MIN(Sale.Price) > 10;
```

Результат наведено на рис. 1.28

В попередньому запиті використовувався новий оператор **HAVING**. Він дозволяє визначити критерій, згідно якого певні групи не включаються в результуючу вибірку, аналогічно тому, як це робить оператор WHERE для окремих записів. Фактично, оператор HAVING дозволяє накласти умову на групи, тому він розміщується після оператора GROUP BY і вказується лише тоді, коли існує останній.

Умову можна визначати і в операторі WHERE, вказуючи на критерій відбору кожного окремого запису, який входить до складу груп, але це робити недоречно. По-перше, падає швидкодія; по-друге, вказуючи умову в операторі WHERE, ви одразу виключаєте відібрані рядки з розгляду будь-якої операції групування. Оператор HAVING, навпаки, починає діяти лише після того, як групи сформовані, визначаючи, які саме групи будуть відображені в результаті.

Зазвичай, оператор HAVING містить функції агрегування, які вказані в списку SELECT, але це необов’язково.

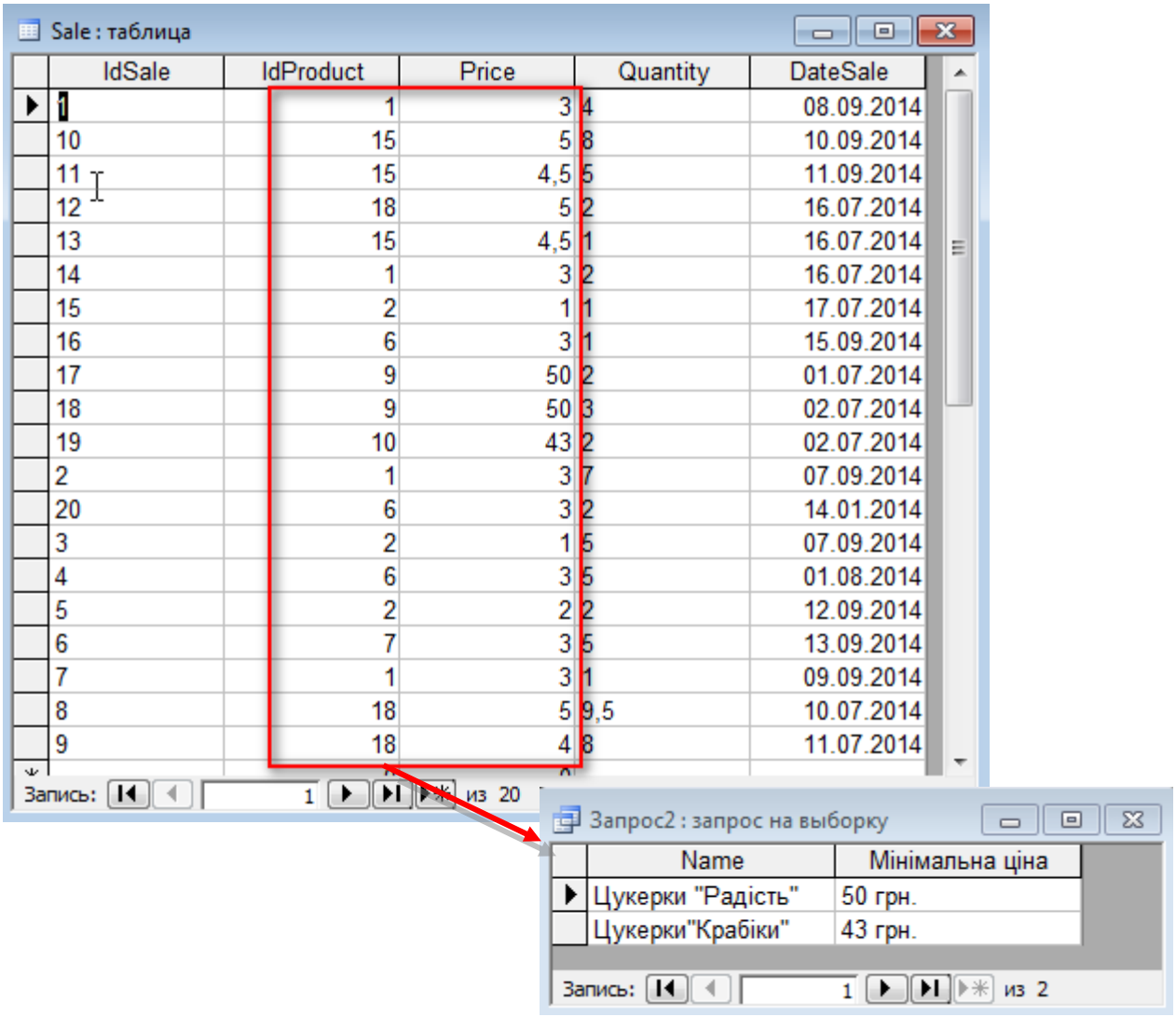


Рис. 1.28. Результат запиту з прикладу 1.62

Приклад 1.63. Вивести кількість товарів кожної категорії, середня ціна поставки яких більше 20 грн.

```
SELECT Category.Name as [Категорія],
       COUNT(Product.IdProduct) as [Кількість]
FROM Product, Delivery, Category
WHERE Product.IdCategory=Category.IdCategory AND
       Product.IdProduct=Delivery.IdProduct
GROUP BY Category.Name
HAVING AVG(Delivery.Price) > 20;
```

Приклад 1.64. Вивести категорії, товари, що до них належать, та загальну суму їх продажу. Умова: товари лише категорій «Кондитерські вироби» та «Фрукти».

```
SELECT Product.Name as [Товар],
       Category.Name as [Категорія],
       Format(SUM(Sale.Price), '## ###.00 грн.')
       as [Сума продаж]
FROM Product, Sale, Category
WHERE Product.IdProduct=Sale.IdProduct AND
        Product.IdCategory=Category.IdCategory
GROUP BY Product.Name, Category.Name
HAVING Category.Name IN ('Кондитерські вироби',
       'Фрукти');
```

1.12. Підзапити

1.12.1.Види та застосування вкладених підзапитів

Стандартом SQL передбачена можливість вкладати запити один до одного, що має велике практичне застосування. В результаті такого вкладення, одні запити можуть управляти іншими. При цьому вводиться поняття підзапиту. **Підзапит** – це запит, який міститься в іншому запиті SQL. Він являє собою повноцінний SELECT вираз, результат виконання якого використовується в іншому запиті. Розміщуватись вони можуть майже в довільному місці SQL-виразу, наприклад, замість одного з імен в списку SELECT, в операторі FROM, при вказанні умови в операторах WHERE або HAVING. Найчастіше зустрічається використання підзапитів при зазначенні умови. Слід відмітити, що самі по собі підзапити не додають жодних функціональних можливостей, але іноді з ними запити стають більш читабельнішими, ніж при складній вибірці.

При використанні підзапитів часто використовують поняття **зовнішнього** та **внутрішнього запиту**. Спочатку виконується підзапит, тобто внутрішній запит, який розміщується, наприклад, в інструкції WHERE, а потім основний, тобто зовнішній запит, який може бути інструкцією SELECT, INSERT, DELETE або UPDATE. При цьому **вкладений підзапит завжди обмежується дужками.**

Приклад 1.65. Написати запит без використання підзапитів, який виводить всю інформацію про товари лише одного постачальника. За звичайних умов ми напишемо:

```
SELECT Product.*
FROM Product, Producer
WHERE Product.IdProducer = Producer.IdProducer
      AND Producer.Name='Хлібзавод№4';
```

Результат подано на рис. 1.29.

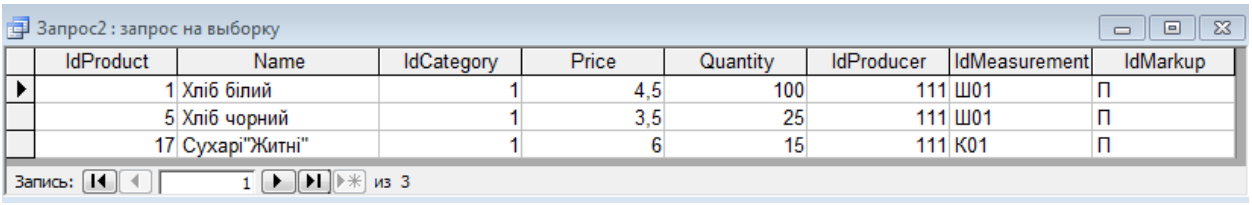


Рис. 1.29. Результат запиту з прикладу 1.65

Приклад 1.66. З використанням підзапитів, попередній запит буде виглядати наступним чином:

```
SELECT *
FROM Product
WHERE IdProducer =
            (SELECT IdProducer
             FROM Producer
              WHERE Name='Хлібзавод№4');
```

Результат буде аналогічний. Розглянемо послідовність процессу під час використання підзапиту.

Спочатку цілісно виконується підзапит, який розміщується в інструкції WHERE. Результатом його виконання буде ідентифікатор, - значення поля Name, що дорівнює “Хлібзавод№4” (рис. 1.30).

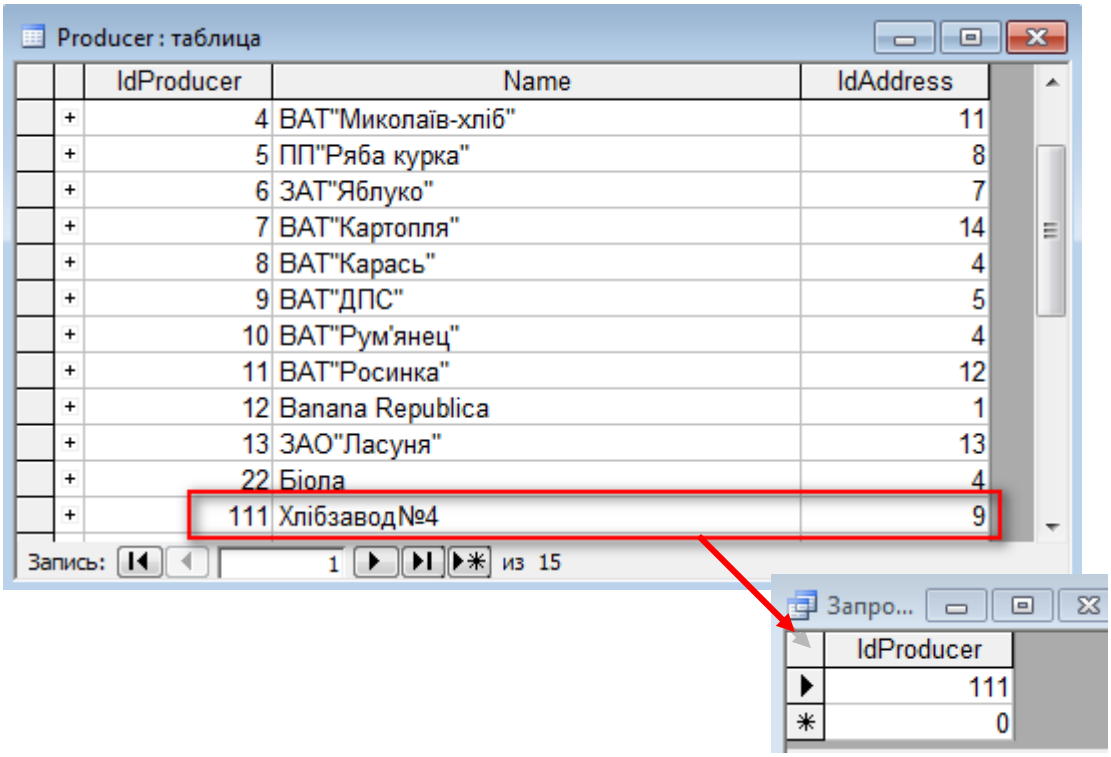


Рис. 1.30. Результат виконання підзапиту з приклада 1.66

Отриманий результат повертається в основний запит і використовується при його виконанні. Тобто зовнішні ключі, які

використовуються для зв'язку з таблицею Producer співставляються з первинним ключем, який є результатом виконання внутрішнього підзапиту. В результаті будуть обрані лише ті товари, які були виготовлені на «Хлібзавод№4».

Розглянемо обмеження при використанні підзапитів. Вони наступні:

1) **Результатом підзапиту повинно бути лише одне значення, причому його тип даних повинен відповідати типу значення, яке використовується в зовнішньому запиті.**

Приклад 1.67. Вивести всю інформацію про товари двох виробників: «Хлібзавод№4» та ВАТ «Росинка».

```
SELECT *
FROM Product
WHERE IdProducer =
            (SELECT IdProducer
             FROM Producer
             WHERE Name='Хлібзавод№4'
              OR   Name='ВАТ "Росинка"');
```

Даний запит призведе до помилки (рис. 1.31), оскільки результатом даного підзапиту є кілька значень, тобто два ідентифікатори, значення полів Name яких рівне "Хлібзавод№4" та ВАТ "Росинка".

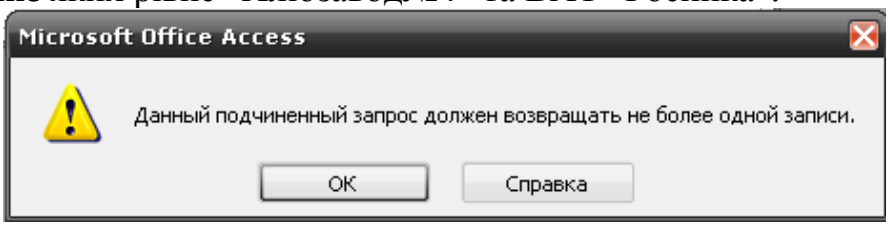


Рис. 1.31. Повідомлення MS Access про помилку

Отже, при використанні запитів, основаних на операторах порівняння або логічних операторах, слід бути дуже уважним, щоб кінцевим результатом підзапиту був лише один запис.

Виходом з цієї ситуації є використання оператора IN, який застосовується для перебору значень результату роботи внутрішнього підзапиту:

Приклад 1.68.

```
SELECT *
FROM Product
WHERE IdProducer IN
            (SELECT IdProducer
             FROM Producer
             WHERE Name='Хлібзавод№4'
                  OR Name='ВАТ "Росинка"');
```

2) **Результатом підзапиту може бути NULL-запис**. В такому випадку результат роботи підзапиту буде оцінений як **UNKNOWN**, що рівносильне FALSE. В результаті попередній запит є вірним навіть без використання оператора IN, але за умови, якщо товарів одного з вказаних виробників або обох в базі даних не існує. В такому випадку підзапит на виході буде поверати один або жодного записів.

   В деяких випадках існує також можливість використання оператора **DISTINCT** для гарантованого отримання одного запису на виході підзапиту.

3) Згідно стандарту ANSI SQL **підзапити є непереміщувані**, тобто наступний запит є вірним:
   Приклад 1.69.

```
SELECT *
FROM Product
WHERE IdProducer =
            (SELECT IdProducer
             FROM Producer
              WHERE Name='Хлібзавод№4' OR
                   Name='ВАТ "Росинка"');
```

Якщо поміняти тіло підзапиту з порівнювальним значенням місцями, повинна згенеруватись помилка:
Приклад 1.70.

```
SELECT *
FROM Product
WHERE (SELECT IdProducer
       FROM Producer
       WHERE Name='Хлібзавод№4' OR
           Name='ВАТ "Росинка"') = IdProducer;
```

Примітка! Зазначене правило не стосується середовища СУБД MS Access, оскільки воно розглядає і перший і другий варіант як однакові.

4) Вкладений запит не можна використовувати в інструкції ORDER BY.
5) При використанні підзапитів для перевірки результату не можна використовувати оператори BETWEEN, LIKE, IS NULL.
6) Якщо підзапит використовується з немодифікованим оператором порівняння, тобто звичаним знаком рівності (=), без використання ключових слів SOME, ANY або ALL, то він не може містити оператори групування GROUP BY та HAVING.

В підзапитах допускається використовувати функції агрегування, оскільки їх результатом є єдине значення. Але і тут слід бути уважним (див. п. 6 зауважень). Розглянемо кілька прикладів:

Приклад 1.71. Скласти запит на отримання інформації про найдорожчий товар, тобто товар з найбільшою ціною продажу:

```
SELECT DISTINCT Product.Name as [Товар],
     Sale.Price &' грн.' as [Ціна]
FROM Product, Sale
WHERE Product.IdProduct = Sale.IdProduct
    AND Sale.Price = (SELECT max(Sale.Price)
                      FROM Sale);
```

Результат подано на рис. 1.32.

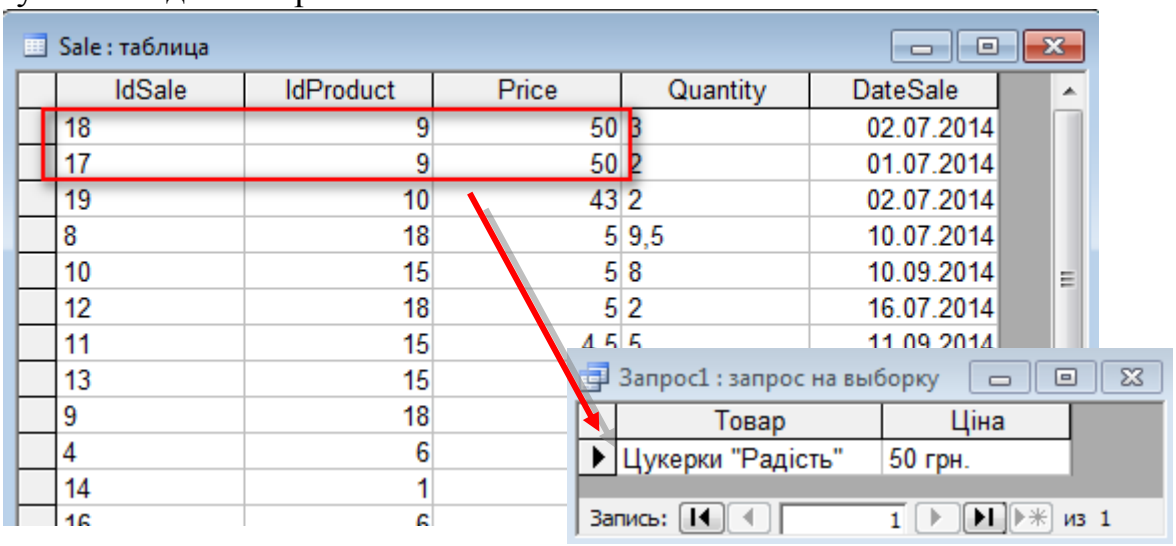


Рис. 1.32. Результат запиту з приклада 1.71

Приклад 1.72. Написати запит, який виводить на екран імена та прізвища всіх постачальників, які поставляли товар в проміжку між 01/06/2014 та поточною датою:

```
SELECT Name as [Постачальник]
FROM Supplier
WHERE IdSupplier IN
                    (SELECT IdSupplier
                     FROM Delivery
                     WHERE DateDelivery BETWEEN
                          #01/06/2014# AND Date());
```

Результат запиту на рис. 1.33.

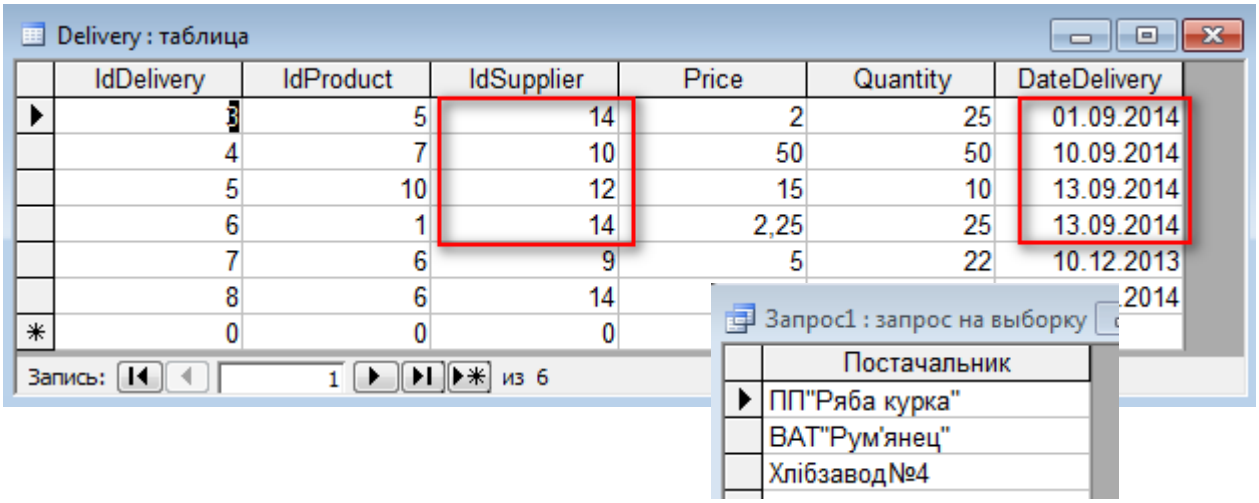


Рис. 1.33. Результат запиту з приклада 1.72

Приклад 1.73. Отримати інформацію про всіх постачальників, які поставляли товар більше, ніж 2 рази. Вибірку відсортувати за назвою виробника:

```
SELECT *
FROM Supplier s
WHERE 2 > ( SELECT COUNT(Delivery.IdSupplier)
             FROM Delivery
              WHERE s.IdSupplier = Delivery.IdSupplier)
ORDER BY 2;
```

Приклад 1.74. Визначити, які товари що поставлялися, були вироблені в м. Київ:

```
SELECT Name as [Товар], Format(Price, '## ###.00 грн.') as [Ціна]
FROM Product
WHERE IdProduct IN ( SELECT DISTINCT IdProduct
                     FROM Delivery
                     WHERE IdSupplier IN (SELECT s.IdSupplier
                                          FROM Supplier s, Address addr
                                          WHERE
                                          s.IdAddress=addr.IdAddress
                                               AND addr.Town = 'Київ'));
```

Як було зазначено, підзапит можна використовувати в виразі FROM зовнішнього запиту. Це дозволяє створити тимчасову таблицю і додати її в запит.

Приклад 1.75. Розглянемо наступний простий запит:

```
SELECT IdProduct, Name, IdCategory
FROM Product
WHERE Price BETWEEN 20 AND 50;
```

Запит з прикладу 1.75. виведе на екран список товарів та ідентифікатори їх категорій, ціна яких знаходиться в межах 20-50 грн. Цей запит можна використати в рамках іншого, щоб отримати додаткову інформацію.

Приклад 1.76. Використаємо запит із попереднього прикладу, щоб вивести список товарів, категорій «М'ясні вироби» та «Фрукти», ціни яких яких знаходяться у вищевказаному діапазоні.

```
SELECT pr.Name as [Товар]
FROM (SELECT IdProduct, Name, IdCategory
      FROM Product
      WHERE Price BETWEEN 20 AND 50) as pr,
      Category as c
WHERE pr.IdCategory = c.IdCategory AND
       c.Name IN ('М'ясні вироби', 'Фрукти');
```

Отже, ми використовуємо підзапит для створення таблиці, яка містить лише три поля: IdProduct, Name та IdCategory. Цій таблиці ми надаємо псевдонім pr. Після цього до створеної таблиці можна звернутись із запитом, як до будь-якої іншої таблиці. В даному випадку, ми використовуємо її, щоб отримати інформацію про товари необхідних категорій.

1.12.2. Оператори EXIST, ANY, SOME, ALL

В попередньому підрозділі було розглянуто в основному однорядкові запити, тобто ті, які повертають один запис. Для того, щоб обробити кілька записів, які поверне підзапит (багаторядковий підзапит), було використано оператор IN. Але він не єдиний. Крім оператора IN, існує ще 4 логічних оператора: **EXISTS**, **ALL**, **ANY** та **SOME**.

Оператор **EXISTS** використовується, коли необхідно визначити наявність значень, які відповідають умові в підзапиті. Якщо дані на виході підзапиту існують, тоді даний оператор поверне значення true, інакше – false. При використанні оператора EXISTS насправді використовують в підзапиті дані зовнішнього запиту. Такий запит іноді називають **зв'язаним** або **корельованим підзапитом**.

Приклад 1.77. Вивести інформацію про всіх постачальників, які коли-небудь поставляли товари в магазин:

```
SELECT *
FROM Supplier
WHERE EXISTS ( SELECT *
                 FROM Delivery
                 WHERE Supplier.IdSupplier =
                                 Delivery.IdSupplier)
ORDER BY 3;
```

Проаналізуємо результат. В підзапиті шукаємо записи таблиці **Delivery**, в яких значення **IdSupplier** співпадає з значенням **Supplier.IdSupplier**. Кожен запис таблиці **Supplier** співставляється з результатом підзапиту і *у випадку існування* (**WHERE EXISTS**) інформація про постачальника додається в результуючу множину.

Інформація, яка повертається підзапитом в підзапит EXISTS або NOT EXISTS немає значення. В зв'язку з цим, інколи повертають довільне константне значення. Все, що необхідно знати, – це сам факт наявності або відсутності записів, які відповідають критерію підзапита.

Приклад 1.78.

```
SELECT *
FROM Supplier
WHERE EXISTS ( SELECT 'X'
               FROM Delivery
               WHERE Supplier.IdSupplier =
                          Delivery.IdSupplier)
ORDER BY 3;
```

Приклад 1.79. *Вивести інформацію про кількість виробників, інформація про яких є в базі даних, але товари яких в магазині відсутні.*

```
SELECT COUNT(*) as [Кількість відсутніх виробників]
FROM Producer
WHERE NOT EXISTS ( SELECT *
                   FROM Product
                   WHERE Product.IdProducer =
                              Producer.IdProducer);
```

Результат подано на рис. 1.34

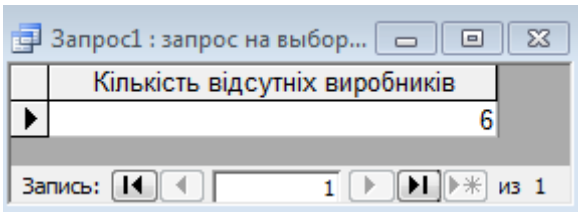


Рис. 1.34. Результат запиту у прикладі 1.79

Оператори **ALL**, **ANY** та **SOME** використовуються для порівняння одного значення з множиною даних, які повертаються підзапитом. Їх можна комбінувати з усіма операторами порівняння і вони можуть включати інструкції GROUP BY та HAVING. Оператори ANY та SOME взагалі є ідентичними і в стандарті ISO вказується, що вони еквівалентні.

Розглянемо для початку дію оператора ANY.

Оператор	Значення
= ANY	Дорівнює довільному значенню, яке повертається підзапитом. Прирівнюється до дії оператора IN і може бути ним замінений
< ANY	Меньше найбільшого значення, яке повертається підзапитом. Це можна трактувати як «меньше будь-якого значення».
> ANY	Більше найменьшого значення, яке повертається підзапитом. Це можна трактувати як «більше будь-якого значення».

Приклад 1.80. Для демонстрації роботи даного оператора *перепишемо запит на отримання інформації про постачальників, які поставляли товари в магазин з використанням оператора ANY:*

```
SELECT IdSupplier, Name
FROM Supplier
WHERE IdSupplier = ANY ( SELECT IdSupplier
                         FROM Delivery)
ORDER BY 2;
```

В даному випадку оператор ANY співставляє всі значення поля **IdSupplier** з таблиці **Delivery** та повертає результат true, якщо **БУДЬ_ЯКЕ (ANY)** значення співпаде із значенням поля **IdSupplier**.

Щодо оператора **SOME (який-небудь)**, то він виведе аналогічний результат.

Приклад 1.81.

```
SELECT IdSupplier, Name
FROM Supplier
WHERE IdSupplier = SOME ( SELECT IdSupplier
                          FROM Delivery)
ORDER BY 2;
```

Різниця між операторами ANY та SOME полягає лише в термінології.

Оператор **ALL** повертає значення *true*, якщо кожне значення, яке буде отримане в результаті роботи підзапиту, відповідає умові зовнішнього запиту. Принцип його дії відображений в наступній таблиці:

Оператор	Значення
> ALL	Більше найбільшого значення, яке повертається підзапитом. Це можна трактувати як «більше за усі значення».
< ALL	Меньше найменьшого значення, яке повертається підзапитом. Це можна трактувати як «менше за усі значень».

Приклад 1.82. В якості прикладу виконаємо запит, в якому поставимо ціллю *довести, що товари категорії "Фрукти" є найбільш продаваємими.*

```
SELECT Product.Name
FROM Product, Sale
WHERE Product.IdProduct = Sale.IdProduct
     AND Sale.Quantity > ALL
                         ( SELECT Sale.Quantity
                           FROM Sale, Product, Category
                           WHERE Sale.IdProduct=Product.IdProduct
                      AND Product.IdCategory=Category.IdCategory
                           AND Category.Name='Фрукти');
```

Підзапит поверне список значень кількості продаж (**Sale.Quantity**) товарів категорії "Фрукти". Потім зовнішній запит шукає кількість продаж товарів, які були більшими, ніж дані, використовуючи для цього вираз **Sale.Qauntity>ALL(кількість продаж)**. Якщо даний запит не поверне жодного запису – це буде підтвердженням того, що товари вказаної категорії є дійсно найбільш продаваємими, інакше виведеться перелік товарів, які продаються більше, ніж вказаної в підзапиті категорії.

Приклад 1.83. Знайти та вивести назви і ціни кондитерських виробів, вартість яких перевищує вартість товарів категорії «Молочні вироби».

```
SELECT DISTINCT Product.Name as [Товар], Format(Product.Price,
      '## ###.00 грн.') as [Ціна]
FROM Product, Category
WHERE Product.Price > ALL (SELECT Product.Price
                           FROM Product, Category
                           WHERE
                      Product.IdCategory=Category.IdCategory
                      AND Category.Name='Молочні')
     AND Product.IdCategory=Category.IdCategory
     AND Category.Name='Кондитерські вироби';
```

Результат подано на рис. 1.35

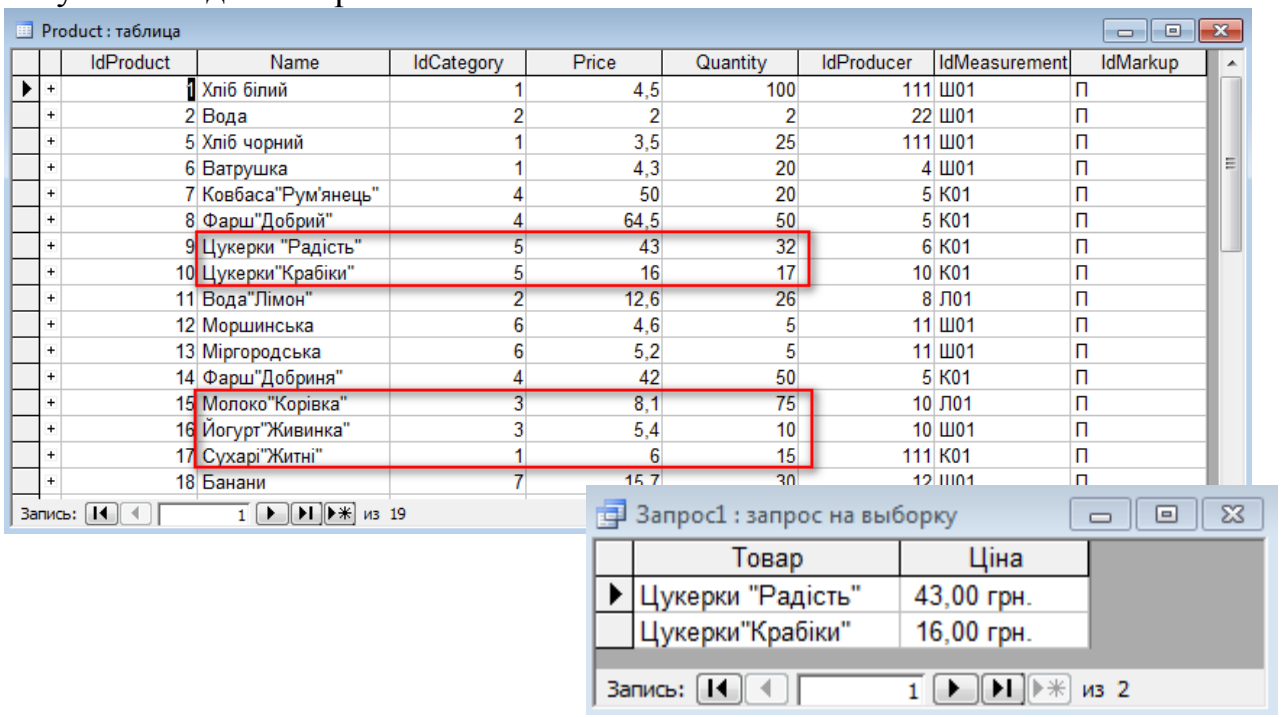


Рис. 1.35. Результат запиту з приклада 1.83

1.13. Об'єднання запитів. Оператори UNION та UNION ALL

Об'єднання – це зв'язування даних, що містяться в двох таблицях або запитах в один результуючий набір. Оскільки об'єднання запитів та таблиць відрізняється, розглянемо окремо один і другий спосіб об'єднання. Розпочнемо з простішого, а саме з об'єднання результатів двох або більше запитів.

Об'єднання запитів здійснюється за допомогою оператора SQL **UNION**. З його допомогою можна об'єднати результати від 2 до 255 результатів запитів в один результуючий набір. В результаті такого об'єднання однакові записи по замовчуванню знищуються, але при наявності ключового слова ALL (тобто при використанні оператора UNION ALL) повертаються всі записи, в тому числі і однакові. Синтаксис оператора UNION наступний:

```
SELECT <список_полів>
[FROM <список_таблиць>]
[WHERE <умова>]
[GROUP BY <список_полів_для_групування>]
[HAVING <умова_на_групу>]
UNION [ALL]
SELECT <список_полів>
[FROM <список_таблиць>]
[WHERE <умова>]
[GROUP BY <список_полів_для_групування>]
[HAVING <умова_на_групу>];
[ORDER BY <умова_сортування>];
```

При використанні оператора UNION **дотримуються наступних правил**:

- кількість, послідовність і типи даних полів в обох списках SELECT повинні співпадати;
- оператори GROUP BY та HAVING використовуються лише в одному запиті;
- жодна з таблиць не може бути відсортована окремо; можна сортувати лише результуючий запит. Тому оператор ORDER BY можна використовувати лише в кінці оператора UNION;
- імена полів результуючої вибірки визначаються списком полів першого оператора SELECT.

*Приклад 1.84. Вибрати всі товари, ціна яких більше 20 грн., **АБО** код категорії товару рівний 1.*

```
SELECT Name
FROM Product
WHERE Price > 20
UNION
SELECT Name
FROM Product
WHERE IdCategory=1;
```

Результат подано на рис. 1.36.

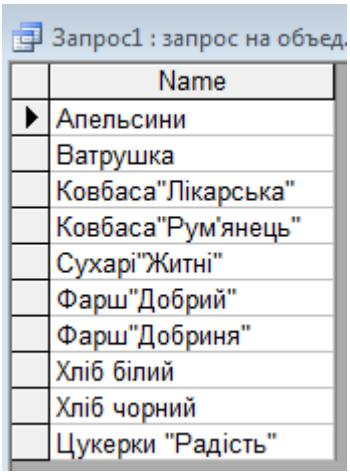


Рис. 1.36. Результат запиту з приклада 1.84

*Приклад 1.85. Вибрати всі товари, ціна яких більше 20 грн., **АБО** які відносяться до категорії "Хлібо-булочні вироби".*

```
SELECT pr.Name as [Товар],
       Format(pr.Price, '## ###.00 грн.') as [Ціна],
       c.Name as [Категорія]
FROM Product pr, Category c
WHEREpr.IdCategory = c.IdCategory AND pr.Price > 20
UNION
SELECT 'назва відсутня',
NULL, Name
FROM Category
WHERE Name ='Хлібо-булочні вироби';
```

Результат подано на рис. 1.37

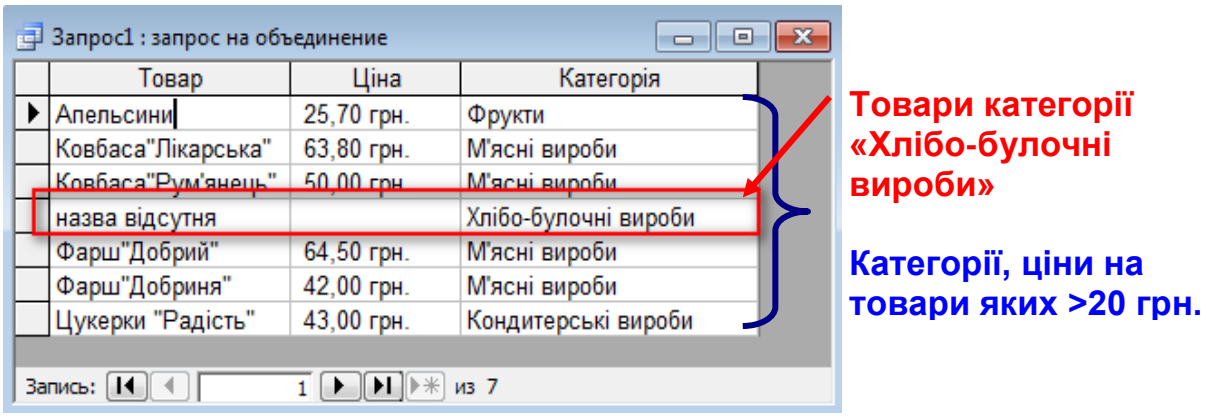


Рис. 1.37. Результат запиту з приклада 1.85

1.14. Об'єднання таблиць. Стандарт SQL

1.14.1. Види об'єднань

Однією з потужних сторін SQL є можливість зв'язувати дані, що розміщуються в окремих таблицях. Це зв'язування здійснюється за допомогою об'єднання таблиць, які вказуються в списку FROM. Разом з цим задається спосіб або тип об'єднання.

СУБД MS Access підтримує лише три **типи об'єднань**:

1. внутрішнє, яке інода називають простим;
2. зовнішнє;
3. самооб'єднання.

Інші СУБД, крім вищеперелічених, підтримують і інші типи об'єднань. Причому, деякі специфічні лише для них. Але ці три типи об'єднань є основними і підтримуються всіма.

Щоб задати спосіб об'єднання, слід скористатись одним з синтаксисів стандарту ANSI: старого стандарту SQL'86 або стандартів SQL2 та вище.

Вони виглядають наступним чином:

```
-- SQL'86
SELECT [таблиця.]поле [,... n]
FROM таблиця
[,таблиця] [, ...]
WHERE умова_об'єднання

-- SQL2 і вище
SELECT [таблиця.]поле [,... n]
FROM таблиця [тип_об'єднання] JOIN таблиця
ON умова_обєднання
```

Як бачимо з опису, синтаксис стандартів відрізняється. Причому, бачимо, що до цього часу ми користувались старим стандартом ANSI SQL, в якому немає можливості вказувати тип об'єднання.

Згідно нового стандарту тип об'єднання вказується ключовим словом при зв'язку двох таблиць. Умова об'єднання, яка тепер вказується після оператора ON є виразом, аналогічним умові відбору, який використовувався в старому стандарті у виразі **WHERE**. Вона задає те, як будуть відноситись між собою записи в двох таблицях. Більшість операцій зв'язування виконуються на базі виразів еквівалентності, таких як **ПолеА = ПолеВ**. Однак умова об'єднання може бути і більша, при цьому всі вирази, які входять в умову об'єднуються за допомогою логічних операторів **AND** або **OR**.

1.14.2. Внутрішні об'єднання. Оператор INNER JOIN

Перший вид об'єднання - **внутрішнє об'єднання**, яке є звичайним об'єднання двох або більше таблиць. Старий стандарт ANSI SQL використовує внутрішній тип об'єднання для зв'язку таблиць.

В стандартах SQL2 і вище, внутрішнє об'єднання здійснюється за допомогою оператора **INNER JOIN**. Використання даного оператора без ключового слова **INNER** також припустимо (СУБД MS Access виключення). В результаті такого об'єднання буде отримана нова таблиця, записи якої задовольняють відповідним умовам. Внутрішні об'єднання повертають дані, якщо знаходять спільну інформацію в обох таблицях.

Приклад 1.86. Відобразити інформацію про товари та їх категорії.

```
SELECT pr.Name as [Товар], c.Name as [Категорія]
FROM Product pr, Category c
WHERE pr.IdCategory = c.IdCategory;
```

Але даний запис використовує старий стандарт ANSI SQL. Перепишемо його згідно вимог стандартів ANSI SQL2 та вище, тобто з використанням оператора **INNER JOIN**.

Приклад 1.87.

```
SELECT pr.Name as [Товар], c.Name as [Категорія]
FROM Product pr INNER JOIN Category c
                ON pr.IdCategory = c.IdCategory;
```

Результат буде однаковий. Фактично, відмінність полягає лише в тому, що зв'язки між таблицями вказуються за допомогою оператора INNER JOIN, а зв'язки за ключовими полями описуються після оператора ON. Всі інші оператори діють так само, як і раніше.

Приклад 1.88. Розглянемо ще один приклад, в якому в зв'язку буде приймати участь більше двох таблиць. *Додамо до запиту з прикладу 1.86. інформацію про виробника товару.* Згідно старого стандарту ANSI SQL такий запит буде виглядати так:

```
SELECT Product.Name as [Товар],
       Category.Name as [Категорія],
       Producer.Name as [Виробник]
FROM Product, Category, Producer
WHERE Product.IdProducer=Producer.IdProducer
    AND Product.IdCategory = Category.IdCategory;
```

Згідно стандарту SQL2 і вище:

Приклад 1.89.

```
SELECT Product.Name as [Товар],
       Category.Name as [Категорія],
       Producer.Name as [Виробник]
FROM Category INNER JOIN
             (Product INNER JOIN Producer ON
              Product.IdProducer=Producer.IdProducer)
          ON Product.IdCategory=Category.IdCategory;
```

Отже, при використанні нового стандарту ANSI SQL2 слід запам'ятати один принцип: **при зв'язку таблиць, вони повинні утворювати суцільний послідовний ланцюг**. В нашому випадку він наступний (рис. 1.37):

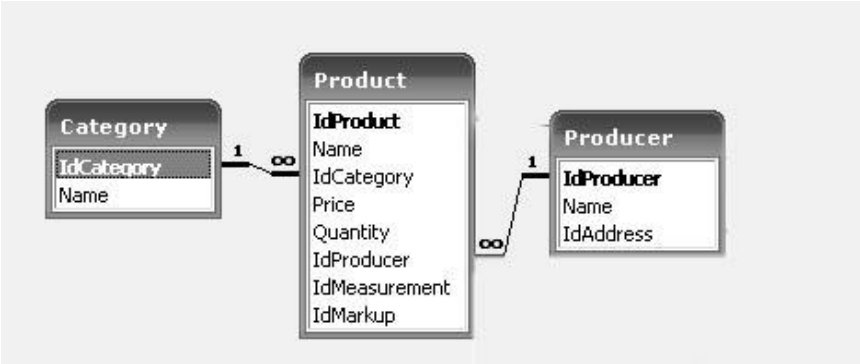


Рис. 1.37. Зв'язки між таблицями у прикладі 1.89

Накладання умови, здійснюється в інструкції WHERE.

Приклад 1.90. Накладемо умову на виведення товарів лише категорії "Бакалія":

```
SELECT Product.Name as [Товар], Category.Name as
      [Категорія], Producer.Name as [Виробник]
FROM Category INNER JOIN (Product INNER JOIN
                     Producer ON
                     Product.IdProducer=Producer.IdProducer)
                ON Product.IdCategory=Category.IdCategory
WHERE Category.Name='Бакалія';
```

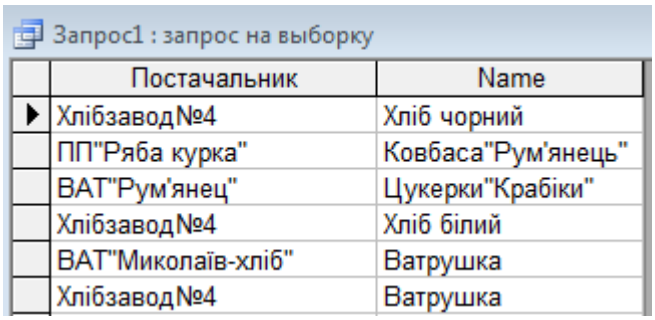
1.14.3. Зовнішнє об'єднання (OUTER JOIN) та його типи: ліве, праве та повне

При використанні оператора INNER JOIN СУБД шукає та виводить записи, які задовольняють критеріям пошуку для двох або кількох таблиць. Але що робити у випадках, коли необхідно знайти записи однієї таблиці, відповідностей яких немає в іншій?

Приклад 1.91. Відобразити інформацію про всі товари та постачальників, що їх поставляли.

```
SELECT Supplier.Name AS Постачальник, Product.Name
FROM Supplier INNER JOIN (Product INNER JOIN
          Delivery ON Product.IdProduct = Delivery.IdProduct)
ON Supplier.IdSupplier = Delivery.IdSupplier;
```

Результат подано на рис. 1.38


Запрос1 : запрос на выборку

Постачальник	Name
Хлібзавод№4	Хліб чорний
ПП"Ряба курка"	Ковбаса"Рум'янець"
ВАТ"Рум'янец"	Цукерки"Крабіки"
Хлібзавод№4	Хліб білий
ВАТ"Миколаїв-хліб"	Ватрушка
Хлібзавод№4	Ватрушка

Рис. 1.38. Результат запиту з приклада 1.91

Як видно з результатів запиту, на екрані відобразились лише ті товари, які поставлялись і про які існує інформація в базі даних. Щоб **відобразити постачальників, інформація про яких присутня в базі даних, не залежно від того, поставляв він вже якийсь товар в магазин чи ні,** потрібно скористатись зовнішнім об'єднанням таблиць.

Зовнішні об'єднання здійснюються за допомогою оператора OUTER JOIN та використовуються у випадку, коли потрібно, щоб запит повертав всі записи з однієї або більше таблиць, незалежно від того, чи мають вони відповідні записи в іншій таблиці. Фактично, вони дозволяють обмежити кількість полів, що повертаються, однієї таблиці, не обмежуючи при цьому їх для іншої таблиці.

Стандарт ANSI виділяє наступні **типи зовнішніх об'єднань** та оператори, що їх здійснюють:

- **LEFT OUTER JOIN. Ліве об'єднання** – записи першої таблиці (зліва) включаються до результуючої таблиці повністю, а з другої таблиці (справа) в результат включаються лише ті, що мають пару в першій таблиці. В якості пари для записів першої таблиці, які не мають пари в іншій використовують пусті (NULL) поля.
- **RIGHT OUTER JOIN. Праве об'єднання** – навпаки.
- **FULL OUTER JOIN. Повне об'єднання –** відображає усі рядки з обидвох таблиць та пов'язує лише ті, які можуть бути пов'язані.

Але СУБД MS Access підтримує лише два перших види зовнішніх об'єднань: ліве і праве.

Приклад 1.92. Для демонстрації роботи зовнішніх об'єднань перепишемо запит з попереднього прикладу таким чином, щоб він *виводив список постачальників, інформація про яких присутня в базі даних, не залежно від того, поставляв він вже якийсь товар в магазин чи ні:*

```
SELECT Supplier.Name, Product.Name
FROM Supplier LEFT OUTER JOIN (Delivery LEFT OUTER JOIN
                        Product ON Product.IdProduct =
    Delivery.IdProduct) ON Supplier.IdSupplier =
    Delivery.IdSupplier;
```

Результат показано на рис. 1.39

Запрос1 : запрос на выборку

Supplier.Name	Product.Name
ВАТ"Картопля"	
ВАТ"Карась"	
ЗАО"Ласуня"	
ВАТ"Миколаїв-хліб"	Ватрушка
ПП"Ряба курка"	Ковбаса"Рум'янець"
ВАТ"Росинка"	
ВАТ"Рум'янец"	Цукерки"Крабіки"
СпортМайстер	
Хлібзавод№4	Хліб чорний
Хлібзавод№4	Хліб білий
Хлібзавод№4	Ватрушка
ВАТ"Яблуко"	

Рис. 1.39. Результат запит уз приклада 1.92

В результаті запиту, якщо постачальник не постачав ще товар в магазин, йому відповідає NULL-значення в полі назви товару, що поставляється.

Приклад 1.93. Відобразити повний список товарів, з інформацією про їх постачальників, незалежно від того відома про них інформація чи ні:

```
SELECT Supplier.Name AS Постачальник, Product.Name AS Товар
FROM Supplier RIGHT OUTER JOIN (Product RIGHT OUTER JOIN
                    Delivery ON Product.IdProduct =
                                Delivery.IdProduct)
              ON Supplier.IdSupplier = Delivery.IdSupplier;
```

Такий запит дозволить одразу виявити товари, інформація про постачальників яких не заповнена. В БД «Магазин» передбачено, щоб такої ситуації не трапилось, адже наявність такої інформації є важливим.

1.14.4. Самооб'єднання таблиць

Аналогічно об'єднанню кількох таблиць, таблицю можна об'єднати саму з собою. Такий вид об'єднання носить назву **самооб'єднання**. Це може знадобитись, коли потрібні зв'язки між рядками однієї і тієї ж таблиці.

Приклад 1.94.1 Вивести інформацію про виробників, назви яких починаються з «ВАТ», тобто форма відповідальності яких Відкрите Акціонерне Товариство:

```
SELECT p1.Name
FROM Producer p1, Producer p2
WHERE p1.IdProducer=p2.IdProducer
      AND p1.Name LIKE 'ЗАТ*';
```

Або:

Приклад 1.94.2

```
SELECT p1.Name
FROM Producer p1 INNER JOIN Producer p2
                    ON p1.IdProducer =p2.IdProducer
WHERE p1.Name LIKE 'BAT*';
```

Результат виконання прикладу на рис. 1.39.

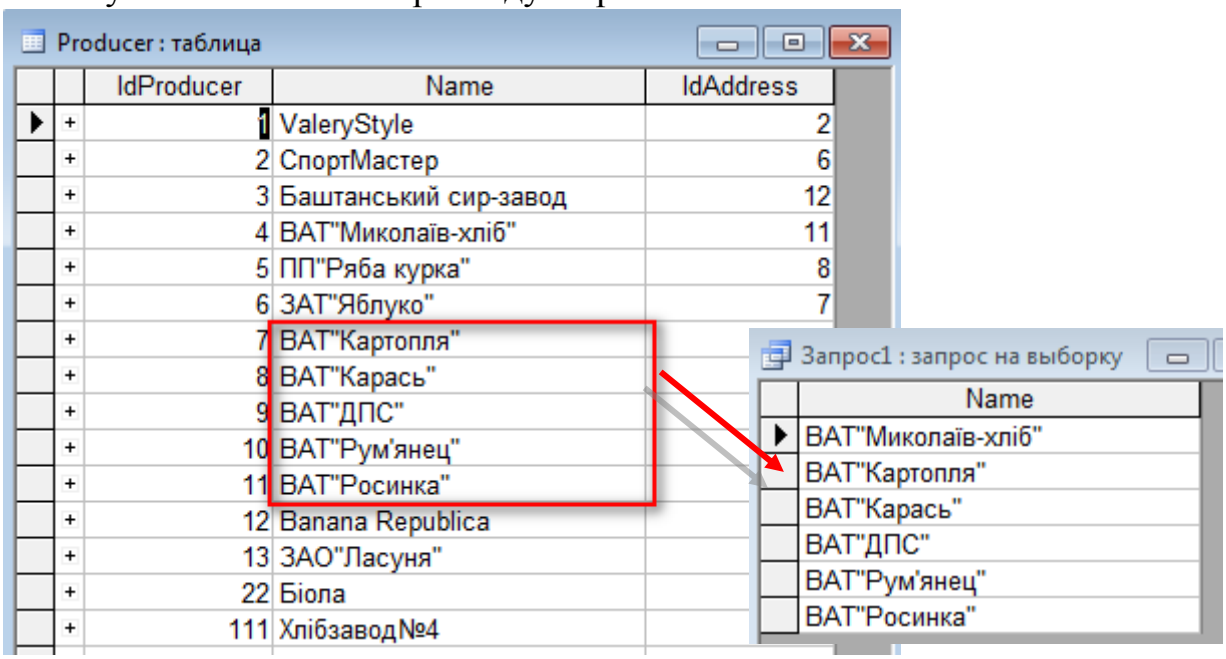


Рис. 1.39. Результат запиту з приклада 1.94.1 або 1.94.2

В такому запиті для таблиці **Producer** визначено два різних псевдоніми, тобто ми повідомляємо саму СУБД, що потрібно мати дві різні таблиці, які повинні містити однакові дані. Після цього вони об'єднуються так само, як і будь-які інші таблиці. А далі отримаємо записи, що задовольняють умові.

Під час оботи з кількома таблицями в запитах ідея самооб'єднання таблиць не повинна викликати великих труднощів.

1.15. Перехресні запити

Перехресні запити є більш складною категорією запитів на вибірку. В них також використовується групування, але на цей раз двовимірне, тобто за записами (рядками) та за полями (стовпчиками). Іншими словами – це вибірка даних, які згруповані за двома критеріями. Як правило, такий тип запитів використовується для формування підсумкових місячних, квартальних або річних звітів за продажами або поставками товарів.

Даний тип запитів притаманний лише MS Access та відрізняється від звичайних групових запитів тим, що в результуючій таблиці перехресного запиту не лише заголовки рядків, але і заголовки стовпчиків **визначаються на основі значень полів, а не їх назв**.

Для створення такого запиту потрібно як мінімум **три елемента**:

1. поле для визначення заголовків рядків;
2. поле, що визначає заголовки стовпчиків;
3. поле для вибору значень, з якими будуть безпосередньо виконуватись розрахунки.

Приклад 1.95. Створити класичний перехресний запит квартальних продаж товарів за 2014 р., взявши за основу попередньо створений багатотабличний запит. ***Базовий запит*** *(назвемо його* ***Report****) повинен відображати загальну вартість для кожного замовленого товару та буде мати наступний вигляд:*

```
SELECT Product.IdProduct as [Код],
       Product.Name as [Товар],
       Sale.DateSale as [Дата продажу],
       Sale.Price*Sale.Quantity as [Вартість]
FROM Product INNER JOIN Sale
            ON Product.IdProduct = Sale.IdProduct
WHERE Sale.DateSale BETWEEN #01/01/2014# AND Date();
```

Сам перехресний запит має свої особливості синтаксису, тому для кращого розуміння, створимо його спочатку за допомогою візарда, а потім проаналізуємо SQL-код, який його створює.

Для того, щоб створити перехресний запит, необхідно обрати на панелі інструментів вікна бази даних кнопку **«Создать»** і в діалоговому вікні вибору типу нового запиту обрати пункт **«Перекрестный запрос»** (рис. 1.96).

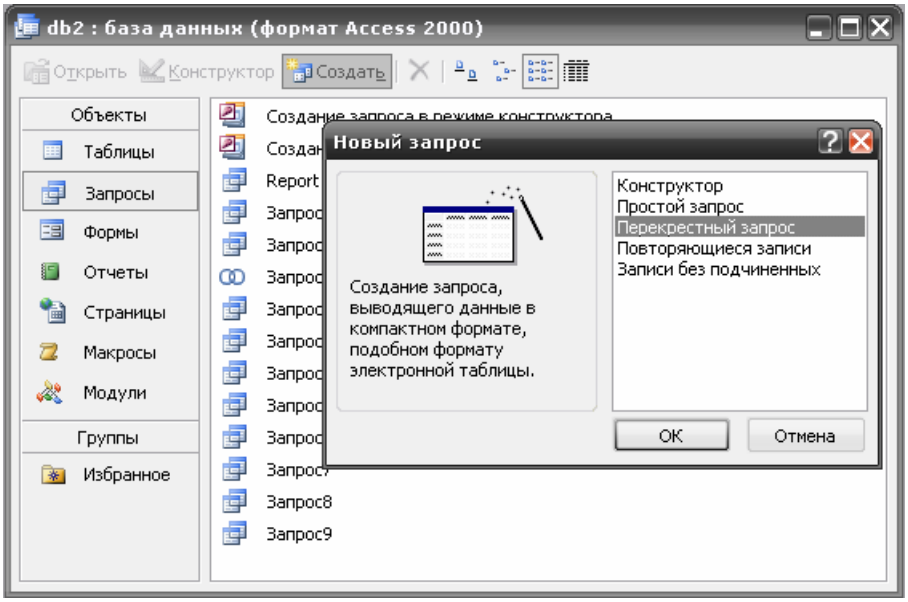


Рис. 1.96. Діалогове вікно вибору режиму створення запиту

Далі візард запропонує здійснити наступні **кроки**:

1. Обрати джерело даних для перехресного запиту. Для цього потрібно обрати в групі опцій **«Показать»** елемент **«Запросы»** і зі списку доступних запитів БД обрати необхідний (рис. 1.97).

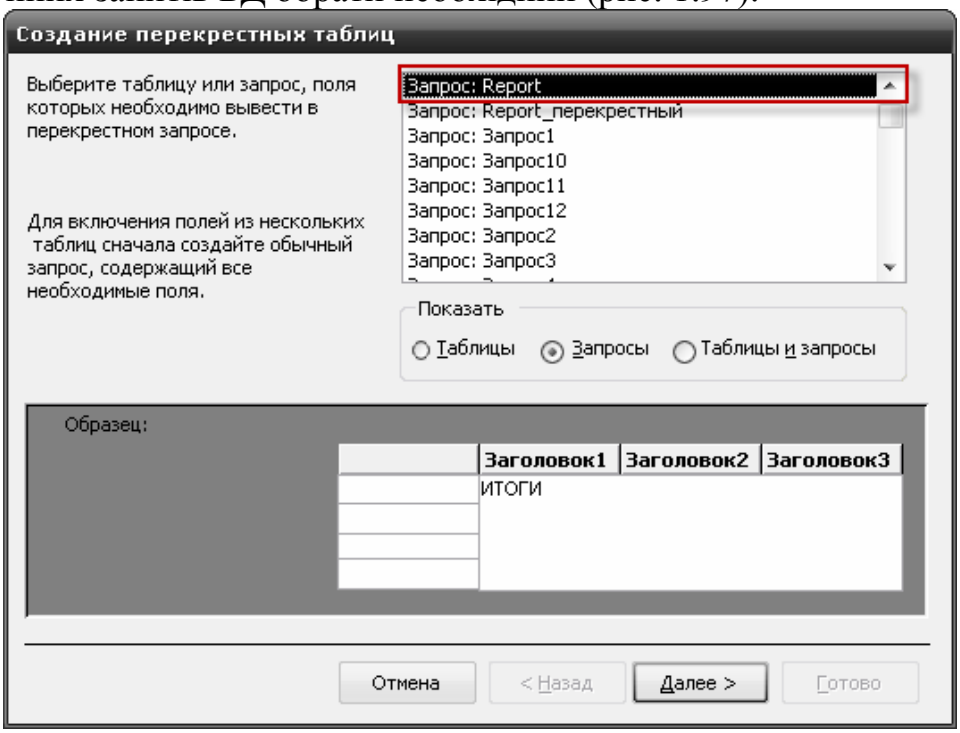


Рис. 1.97. Вибір запиту у вікні створення перехресного запиту

2. На наступному кроці обрати зі списку **«Доступные поля»** ті поля, значення яких будуть використовуватись як заголовки рядків перехресного запиту. Необхідні поля перенести за допомогою кнопок в список **«Выбранные поля»**. В нашому випадку, необхідно сформувати звіт за товарами, тому слід обрати поле «Товар» (рис. 1.98).

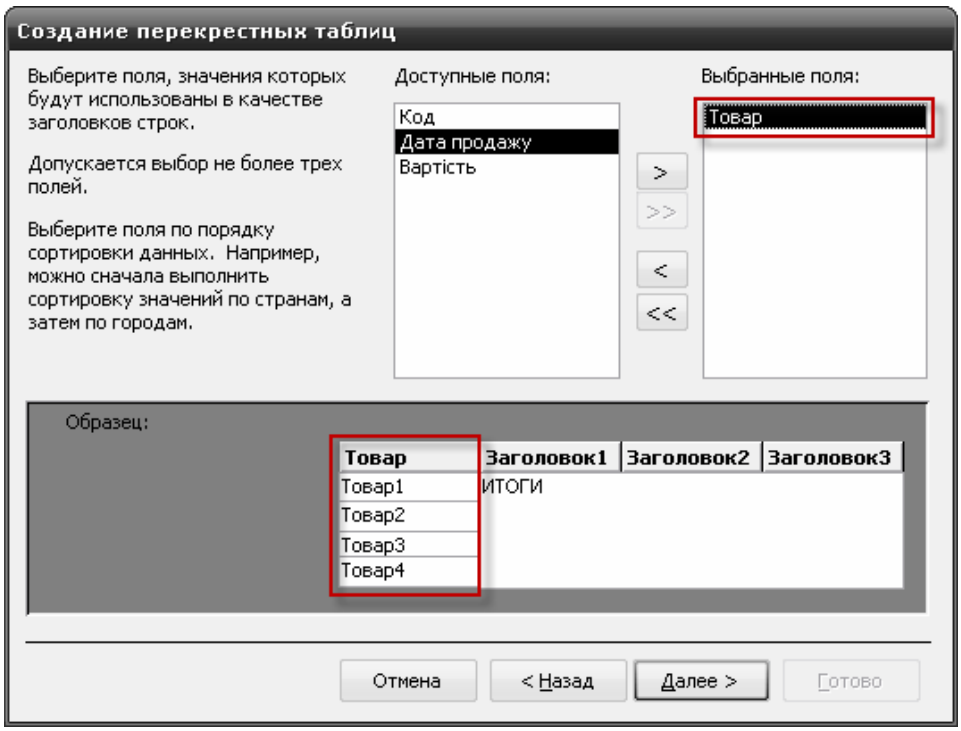


Рис. 1.98. Вибір поля запиту у вікні створення перехресного запиту

3. Далі необхідно обрати поле, значення якого буде використане в якості заголовка стовпчиків. Оскільки квартальний звіт передбачає, що підрахунок продажу товарів буде здійснюватись за кожною датою в межах кварталу, то слід обрати поле «Дата продажу» (рис. 1.99).

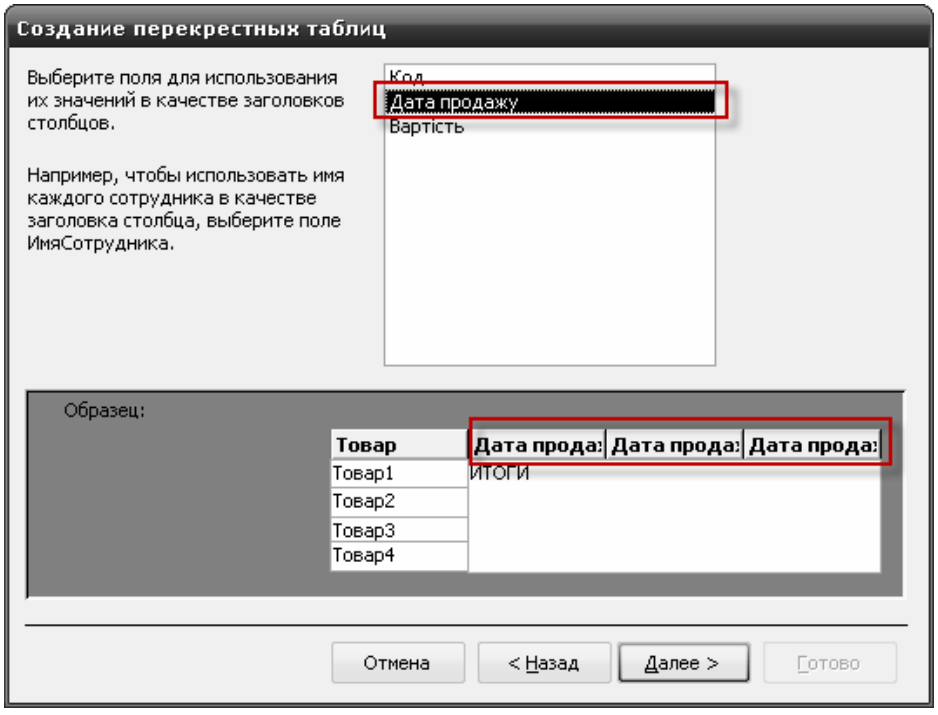


Рис. 1.99. Вибір заголовків стовпчиків запиту у вікні створення перехресного запиту

4. Після цього потрібно обрати інтервал групування для стовпчиків, тобто для дат продажу. Оскільки звіт в нас квартальний, - обираємо «Квартал»

(рис. 1.100).

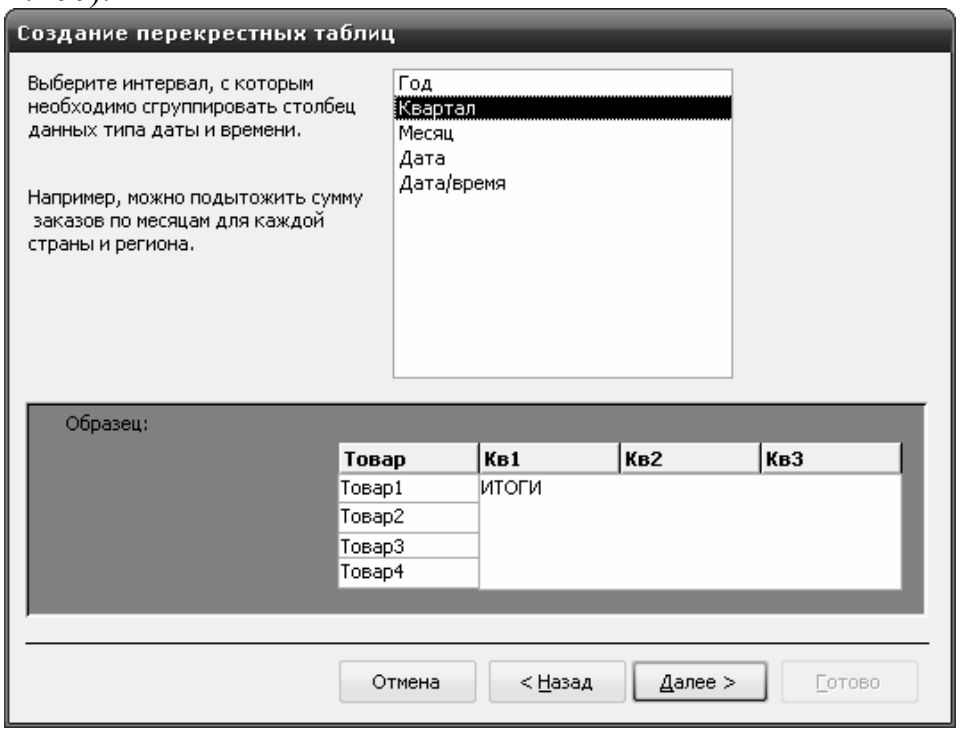


Рис. 1.100. Вибір інтервалу групування для стовпчиків у вікні створення перехресного запиту

5. Останнім кроком є вибір підсумкової операції, яку необхідно застосувати для обробки даних поля «Вартість» (в нашому випадку). Дана операція фактично буде відображати ту інформацію, яка нас цікавить: загальну суму продажів в розрізі кварталу, середню вартість тощо. Нас цікавить сума, тому в списку **«Функции»** слід обрати відповідний пункт (рис. 1.101).

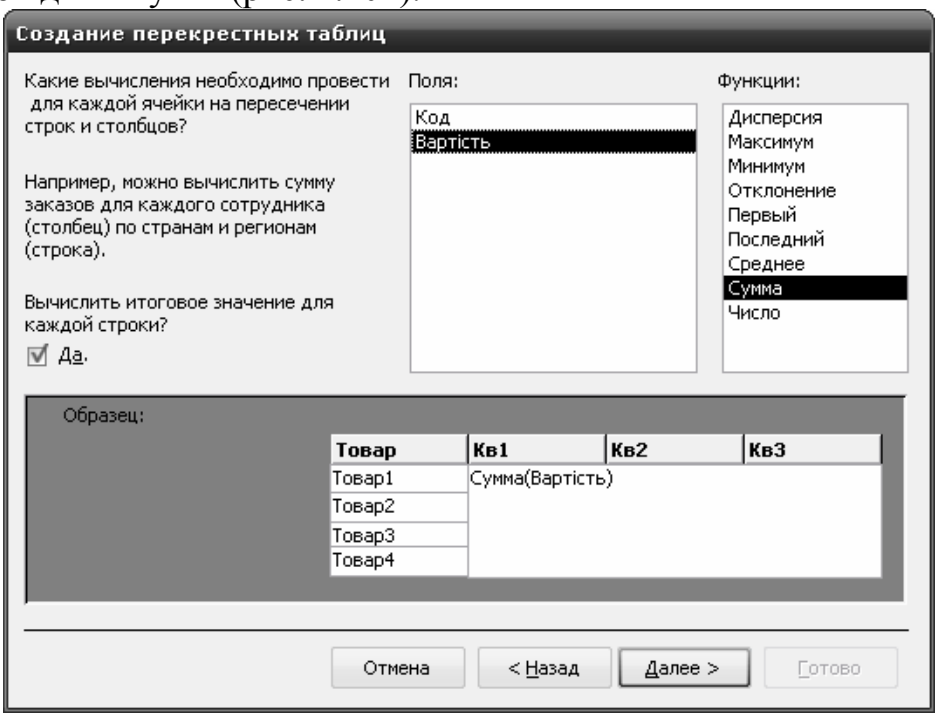


Рис. 1.101. Вибір підсумкової операції для обробки даних у вікні створення перехресного запиту

Результат буде мати наступний вигляд, поданий на рис. 1.102

Report_перекрестный : перекрестный запрос

Товар	Итоговое значение Вартість	Кв1	Кв3
Банани	89,5		89,5
Ватрушка	24	6	18
Вода	10		10
Ковбаса"Рум'янець"	15		15
Молоко"Корівка"	67		67
Хліб білий	42		42
Цукерки "Радість"	250		250
Цукерки"Крабіки"	86		86

Запись: 1 из 8

Рис. 1.102. Результат створення перехресного запиту

Як бачимо з результату, в полі «Товар» відображається список товарів, які були продані упродовж 2014 року. Поле «Итоговое значение Вартість» (його можна потім перейменувати для зручності) містить дані про те, на яку суму за цей період кожного товару було продано, а в сусідніх полях («Кв1», «Кв2» (не існує в БД), «Кв3») ця інформація відображена в розрізі кварталів.

Створення перехресного запиту за допомогою візарда, крім основної своєї переваги, – наочності та зручності, має більш суттєві **недоліки**. До них слід **віднести**:

- працювати можна лише з одним звітом чи таблицею;
- неможна сортувати результуючу таблицю за значеннями, які містяться в полях, оскільки в більшості випадків одночасне впорядкування даних в полях по всім записам неможливе. Але ви можете задати сортування для заголовків записів (рядків);
- в процесі створення запиту неможна вказувати додаткові умови відбору.

В режимі SQL даний запит матиме наступний вигляд:

Приклад 1.95.1.

```
TRANSFORM Sum(Report.Вартість) AS [Sum-Вартість]
SELECT Report.Товар, Sum(Report.Вартість)
       AS [Итоговое значение Вартість]
FROM Report
GROUP BY Report.Товар
PIVOT "Кв" & Format([Дата продажу],"q");
```

Зверніть увагу на формування самого запиту та використання ключових слів **TRANSFORM** і **PIVOT**, які, не є зарезервованими в стандарті ANSI SQL. Операція TRANSFORM в даному запиті дозволяє визначити дані, які містяться в результуючій таблиці, а в операції PIVOT задаються заголовки стовпчиків.

Повний синтаксис оператора **TRANSFORM**:

```
TRANSFORM функція_агрегування
SELECT вибірка
PIVOT поля_для_заголовків
       [IN (значення1 [, значення2 [,...]])];
```

Функція агрегування біля оператора **TRANSFORM** дозволяє обробити відібрані дані. Оператор **SELECT**, який записується наступним, вказує поля, які будуть використовуватись в якості назв записів (рядків). Він **обов'язково** повинен містити вираз **GROUP BY**, який вказує на спосіб групування записів. Крім виразу **GROUP BY**, оператор **SELECT** повноцінно може містити і інші конструкції (**WHERE**, **ORDER BY** тощо) та підзапити.

Поряд з оператором **PIVOT** вказуються імена полів або вирази, які будуть використовуватись в якості заголовків стовпчиків в результуючій таблиці. Наприклад, якщо вказати назви місяців, то результуючий запит буде містити дванадцять стовпчиків. Цей список полів можна обмежити, створивши заголовки з набору констант, які перераховуються в виразі IN. Іншими словами, значення, які не входять в набір, будуть відфільтровані.

Приклад 1.96. Написати перехресний запит на формування квартального звіту за продажами товарів, без використання проміжного запиту. Відсортувати вибірку.

```
TRANSFORM Sum(Sale.Price*Sale.Quantity) as [Вартість]
SELECT Product.Name as [Товар],
     Sum(Sale.Price*Sale.Quantity)
                    as [Итоговое значение Вартість]
FROM Product INNER JOIN Sale
               ON Product.IdProduct = Sale.IdProduct
WHERE Sale.DateSale BETWEEN #01/01/2009# AND Date()
GROUP BY Product.Name
ORDER BY Product.Name
PIVOT "Кв" & Format(Sale.DateSale,"q");
```

1.16. Запити з параметрами

Існують випадки, коли значення критеріїв запиту невідомі наперед, тобто не є константними. Наприклад, необхідно вивести список поставлених або проданих товарів на певну дату, але ця дата кожен раз при формуванні звіту змінюється. В такому випадку, можна постійно писати новий запит, що не досить зручно, або ж створити один **параметричний запит**, який буде запитувати користувача значення необхідного критерію (-ів), в залежності від якого (-их) буде виведений результат.

Отже, **параметричний запит** або **запит з параметрами** – це спеціальний інтерактивний тип запиту, який перед виконанням виводить діалогове вікно з проханням ввести один або кілька параметрів, необхідних для його роботи. Ці параметри фактично дозволяють користувачу накладати умови відбору записів.

Приклад 1.97. Створитиз запит, який виводить на екран список товарів, необхідної користувачу категорії:

```
SELECT Product.Name as [Товар],
       Category.Name as [Категорія]
FROM Category INNER JOIN Product
       ON Category.IdCategory = Product.IdCategory
WHERE Category.Name=[Введіть категорію];
```

Зверніть увагу на умову: `Category.Name=[Введіть категорію]`. Для того, щоб вказати, що значення критерію відбору буде отримане від користувача, в конструкції `WHERE` замість конкретного значення вказуються **квадратні дужки**. Якщо всередині них вказати текст, то він буде підказкою користувачу на те, що необхідно ввести.

Після запуску вказанного запиту, з'являється діалогове вікно з проханням ввести категорію, список товарів якої необхідно вивести на екран (рис. 1.103).

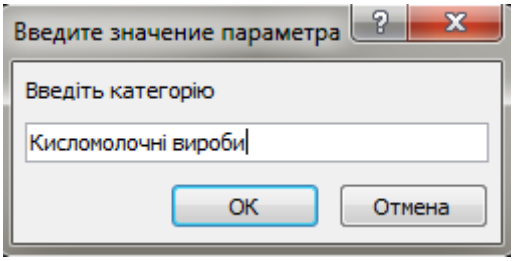


Рис. 1.103. Вікно для введення параметру запиту

Після натиснення на кнопку OK, буде сформований список товарів введеної категорії, тобто молочні продукти (рис. 1.104).

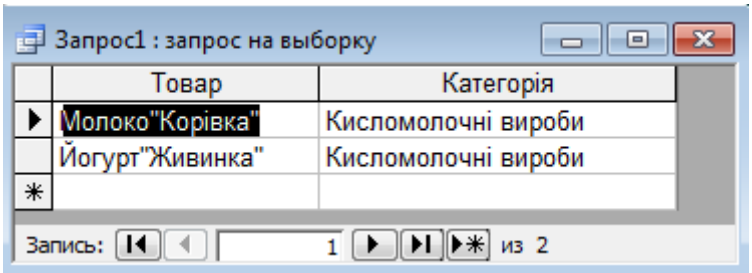


Рис. 1.104. Результат виконання запиту з прикладу 1.97

Приклад 1.98. Відібрати товари, які були поставлені в проміжку певних дат.

```
SELECT Product.Name as [Товар],
       Supplier.Name as [Постачальник], Delivery.
       DateDelivery as [Дата постаки]
FROM Supplier INNER JOIN (Product INNER JOIN Delivery
                  ON Product.IdProduct = Delivery.IdProduct)
         ON Supplier.IdSupplier = Delivery.IdSupplier
WHERE Delivery.DateDelivery BETWEEN [Початкова дата]
                                    AND [Кінцева дата];
```

Параметризовані запити такого вигляду притаманні лише СУБД MS Access. В більшості інших СУБД цей вид запитів представлений окремими об'єктами бази даних, такими як збережені процедури та функції.

1.17. Запити на створення таблиці. Конструкція TOP

В результаті роботи даних запитів, здійснюється вибірка даних з однієї або кількох таблиць і її результат зберігається в новій таблиці. Запити на створення нової таблиці використовуються для створення архівних копій таблиць, копій для експорта в іншу базу даних або ж як джерело даних для звіту. Наприклад, звіт про щомісячні продажі за регіонмих можна зробити на основі одного і того ж запиту.

Для створення такого виду запитів, в синтаксисі оператора SELECT існує опція **INTO**. Дана опція вказується після списку SELECT, але перед вказанням списку таблиць, з яких буде отримана інформація (конструкцією FROM):

```
SELECT  [ALL | DISTINCT]
              { * | поле_для_вибірки [, поле_для_вибірки] }
[INTO ім'я_нової_таблиці
[IN зовнішня_база_даних] ]
FROM { таблиця | представлення} [as] [псевдонім]
      [, {таблиця | представлення} [as] [псевдонім] ]
[WHERE умова]
[GROUP BY [ALL] вираз_групування]
[HAVING умова_на_групу]
[ORDER BY імя_поля | номер_поля [ ASC | DESC ] ];
```

Необов'язкова опція **IN** дозволяє вказати БД, в якій буде розташована новостворена таблиця. За замовчуванням, такою базою є поточна, тобто нова таблиця буде збережена в активній БД.

Структура нової таблиці буде відповідати структурі списку SELECT, і тому для уникнення проблем щодо ідентифікації полів новоствореної таблиці бажано вказувати псевдоніми. Крім того, варто відмітити, що при створенні нової таблиці її поля успадковують лише тип і розмір джерела, інші

ж властивості (індекси, первинні ключі тощо) потрібно створювати самостійно.

Приклад 1.99. Створити запит з метою створення ***нової таблиці****, що містить інформацію про назви товарів, їх ціни та категорії.*

```
SELECT Product.Name as [NameProduct],
       Format(Product.Price, '## ###.00 грн.') as [Price],
       Category.Name as [Category]
INTO ProductEx
FROM Product INNER JOIN Category
             ON Product.IdCategory = Category.IdCategory;
```

В результаті роботи даного запиту утвориться нова таблиця **«ProductEx»**, що містить вищезазначені дані (назву товару, його ціну та категорію). Імена полів новоствореної таблиці носять назву псевдонімів полів при написанні запиту, тобто **NameProduct, Price** та **Category** (рис. 1.105).

ProductEx : таблиця

NameProduct	Price	Category
Хліб білий	4,50 грн.	Хлібо-булочні вироби
Вода	2,00 грн.	СокіВоди
Хліб чорний	3,50 грн.	Хлібо-булочні вироби
Ватрушка	4,30 грн.	Хлібо-булочні вироби
Ковбаса"Рум'янець"	50,00 грн.	М'ясні вироби
Фарш"Добрий"	64,50 грн.	М'ясні вироби
Цукерки "Радість"	43,00 грн.	Кондитерські вироби
Цукерки"Крабіки"	16,00 грн.	Кондитерські вироби
Вода"Лімон"	12,60 грн.	СокіВоди
Моршинська	4,60 грн.	Бакалея
Міргородська	5,20 грн.	Бакалея
Фарш"Добриня"	42,00 грн.	М'ясні вироби
Молоко"Корівка"	8,10 грн.	Кисломолочні вироби
Йогурт"Живинка"	5,40 грн.	Кисломолочні вироби
Сухарі"Житні"	6,00 грн.	Хлібо-булочні вироби

Рис. 1.105. Результа запиту з приклада 1.99.

До новоствореної таблиці **ProductEx** можна написати довільний запит і зміни значень в її записах жодним чином не вплинуть на таблиці **Product** та **Category**.

Зверніть увагу на піктограму нашого запиту в списку запитів БД. (рис. 1.106) Вона набула вигляду таблички зі знаком оклику, вказуючи на те, що це запит не на вибірку, а на створення нової таблиці.

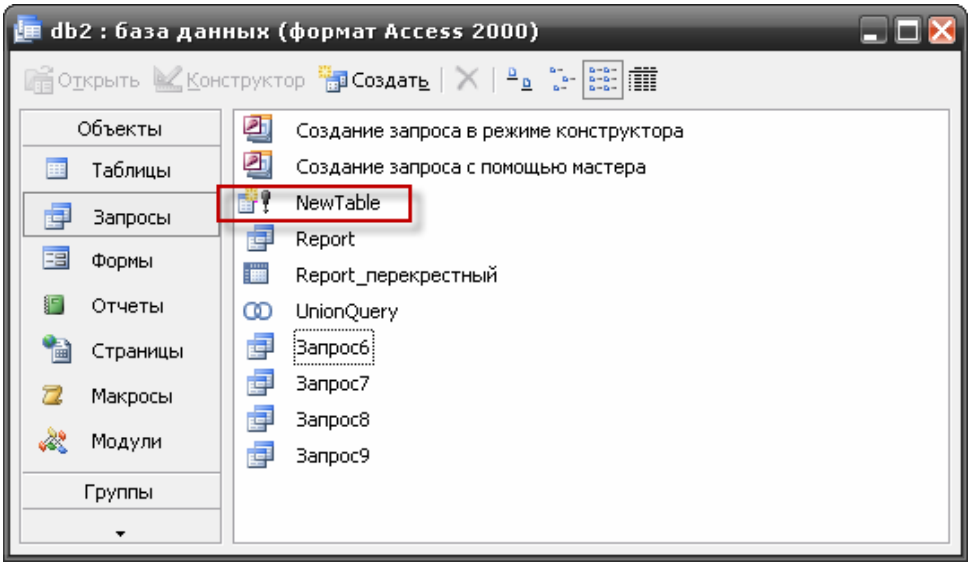


Рис. 1.106. Вікно БД зі списком запитів

Запити на створення таблиці підтримуються не усіма СУБД, кожна з тих, що такі запити підтримує, здійснює його різними засобами. Тому, перед тим як написати запит на створення таблиці, потрібно ознайомитися з документацію по SQL для необхідної СУБД.

Конструкція **TOP** в синтаксисі оператора SELECT дозволяє обрати перших n-записів результуючої вибірки. З даною опцією синтаксис оператора SELECT набуває наступного вигляду:

```
SELECT [ALL | DISTINCT]
      [TOP n] { * | поле_для_вибірки [,
      поле_для_вибірки] } [INTO ім'я_нової_таблиці
[IN зовнішня_база_даних] ]
FROM { таблиця | представлення} [as] [псевдонім]
      [, {таблиця | представлення} [as] [псевдонім] ]
[WHERE умова]
[GROUP BY [ALL] вираз_групування]
[HAVING умова_на_групу]
[ORDER BY імя_поля | номер_поля [ ASC | DESC ] ];
```

Приклад 1.100. Запит, який буде містити інформацію про найчастіше поставляємий товар, тобто товар, замовлення якого було здійснено найбільшу кількість разів.

```
SELECT TOP 1 Product.Name AS [Товар],
      Count(Delivery.IdProduct) AS [Кількість поставок]
FROM Product INNER JOIN Delivery
                  ON Product.IdProduct =
Delivery.IdProduct
GROUP BY Product.Name
ORDER BY 2 DESC;
```

Для отримання такої інформації, спочатку створюється запит на вибірку, в якому підрахується скільки разів був поставлений кожен товар. Після цього результуючий запит сортується і на екран виводиться перший (верхній) запис (TOP 1) (рис. 1.107).

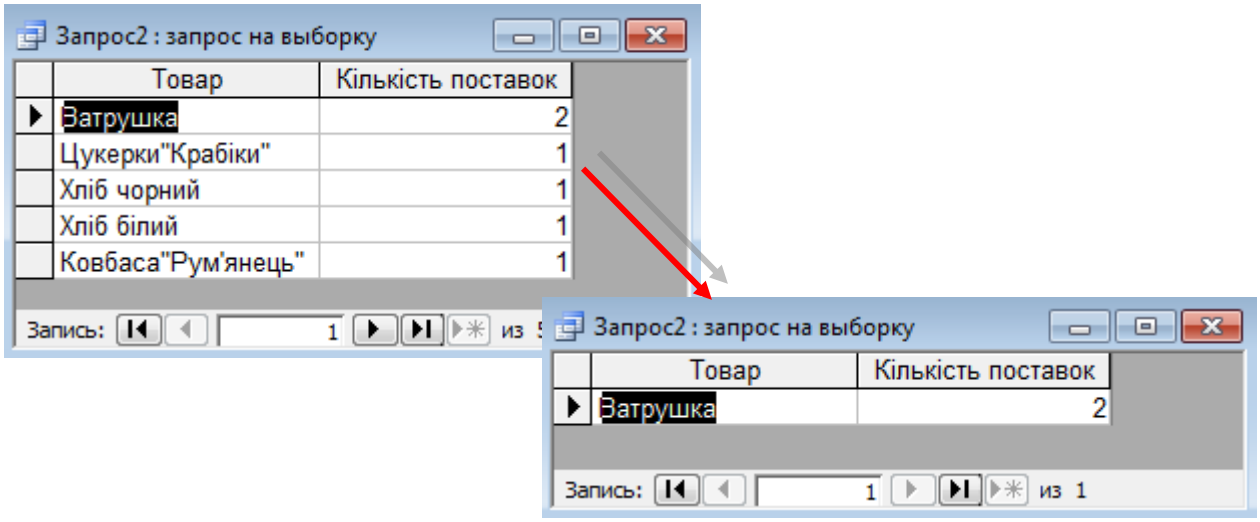


Рис. 1.107. Результа запиту з приклада 1.100.

<u>Приклад 1.101.</u> Результат виконаних вище дій можна зберегти в окрему таблицю, яка буде містити звітну інформацію про три найчастіше поставляємих товари.

```
SELECT TOP 3 Product.Name AS [Товар],
            Count(Delivery.IdProduct) AS [Кількість поставок]
INTO BestDelivery
FROM Product INNER JOIN Delivery
          ON Product.IdProduct = Delivery.IdProduct
GROUP BY Product.Name
ORDER BY 2 DESC;
```

1.18. Модифікація даних засобами DML. Запити на додавання, видалення та поновлення

Команди категорії **DML (Data Manipulation Language – Мова Маніпулювання Даними)** дозволяють маніпулювати даними об'єктів БД. До неї входять оператори:

1. Вставки записів – **INSERT**.
2. Видалення записів – **DELETE**.
3. Зміни значень в існуючих записах таблиці - **UPDATE**.

1.18.1. Оператор INSERT

Оператор SQL **INSERT** використовується для додавання записів в таблиці. Різні варіанти оператора **INSERT** дозволяють додавати в таблицю як один, так і множину записів. Це можна здійснювати як шляхом введення константних значень, так і методом зчитування даних з іншої таблиці. В кожному з перелічених випадків **необхідно враховувати структуру таблиці**:

- кількість полів;
- тип даних кожного поля;
- імена полів, в які вносяться дані;
- обмеження і властивості полів.

Синтаксис оператора **INSERT** наступний:

```
INSERT [INTO] назва_таблиці [( поле1 [, поле2 [,...]] )]
VALUES ( DEFAULT | 'значення1' [, 'значення2' [,...]] );
```

Згідно даного синтаксису у вказану таблицю додається запис зі значеннями полів, вказаними в переліку фрази VALUES (значення), причому і-е значення відповідає і-му полю в списку полів (поля, не вказані в списку, заповнюються NULL-значеннями). Якщо в списку VALUES вказані всі поля модифікуємої таблиці і порядок їх переліку відповідає порядку полів в описі таблиці, то список полів при INTO можна проігнорувати.

Приклад 1.102. Додати до списку категорій нову категорію товарів:

```
INSERT INTO Category
VALUES (11, 'Рибні продукти');
```

Таким чином, значення вставляються у всі поля таблиці **Category** у відповідності з їх фізичним порядком. Якщо потрібно вставити значення тільки в деякі поля, тоді їх потрібно явно вказувати.

Приклад 1.103. Наприклад, ідентифікатор товару вказувати недоречно, він повинен генеруватись автоматично, слід вказувати лише назву категорії. В такому разі запит на добавлення записів слід переписати:

```
INSERT INTO Category (Name) VALUES ('Соки та води');
```

Приклад 1.104. Випадок, якщо таблиця має зовнішній ключ. До виробників продукції потрібно додати дані про нового виробника. Таблиця «Producer» має два основних поля: Name (назву) та IdAddress (зовнішній ключ, який містить посилання на повну адресу виробника). Запит на вставку даних буде мати наступний вигляд:

```
INSERT INTO Producer (Name, IdAddress)
       VALUES ('ПП Постовалов І.В.', 3);
```

Для **додавання кількох записів** шляхом вибірки даних з іншої таблиці використовується модифікований синтаксис оператора INSERT:

```
INSERT [INTO] назва_таблиці
     [IN зовнішня_база_даних]
                [( поле1 [, поле2 [,...]] )]
SELECT ( поле1 [, поле2 [,...]] )
FROM список_таблиць;
```

Вираз **IN** дозволяє вказати назву зовнішньої бази даних, у вказану таблицю якої будуть вставлені нові дані. Те, які саме дані будуть додані, визначається оператором SELECT.

Приклад 1.105. Припустимо, в нас існує таблиця **SupplierArchive**, яка містить копію даних про постачальників.

```
INSERT INTO SupplierArchive SELECT *
FROM Supplier;
```

Приклад 1.106. Якщо необхідно накласти умову на вставку даних, наприклад, в нову таблицю **SupplierUkr** потрібно вставити дані про усіх постачальників з України, тоді запит на додавання перепишеться:

```
INSERT INTO SupplierUkr (Name, IdAddress)
SELECT Supplier.Name, Supplier.IdAddress
FROM Country INNER JOIN (Address INNER JOIN Supplier
                ON Address.IdAddress = Supplier.IdAddress)
              ON Country.IdCountry = Address.IdCountry
WHERE Country.Name='Україна';
```

Список полів при `INTO` та `SELECT` в даному випадку є обов'язковим, оскільки відібрані дані про постачальників можуть мати невпорядковані ідентифікатори (наприклад, 5 і 10), а нова таблиця може мати свої значення порядкових номерів для нових даних, а також вимагає, щоб вони йшли за порядком (рис. 1.108).

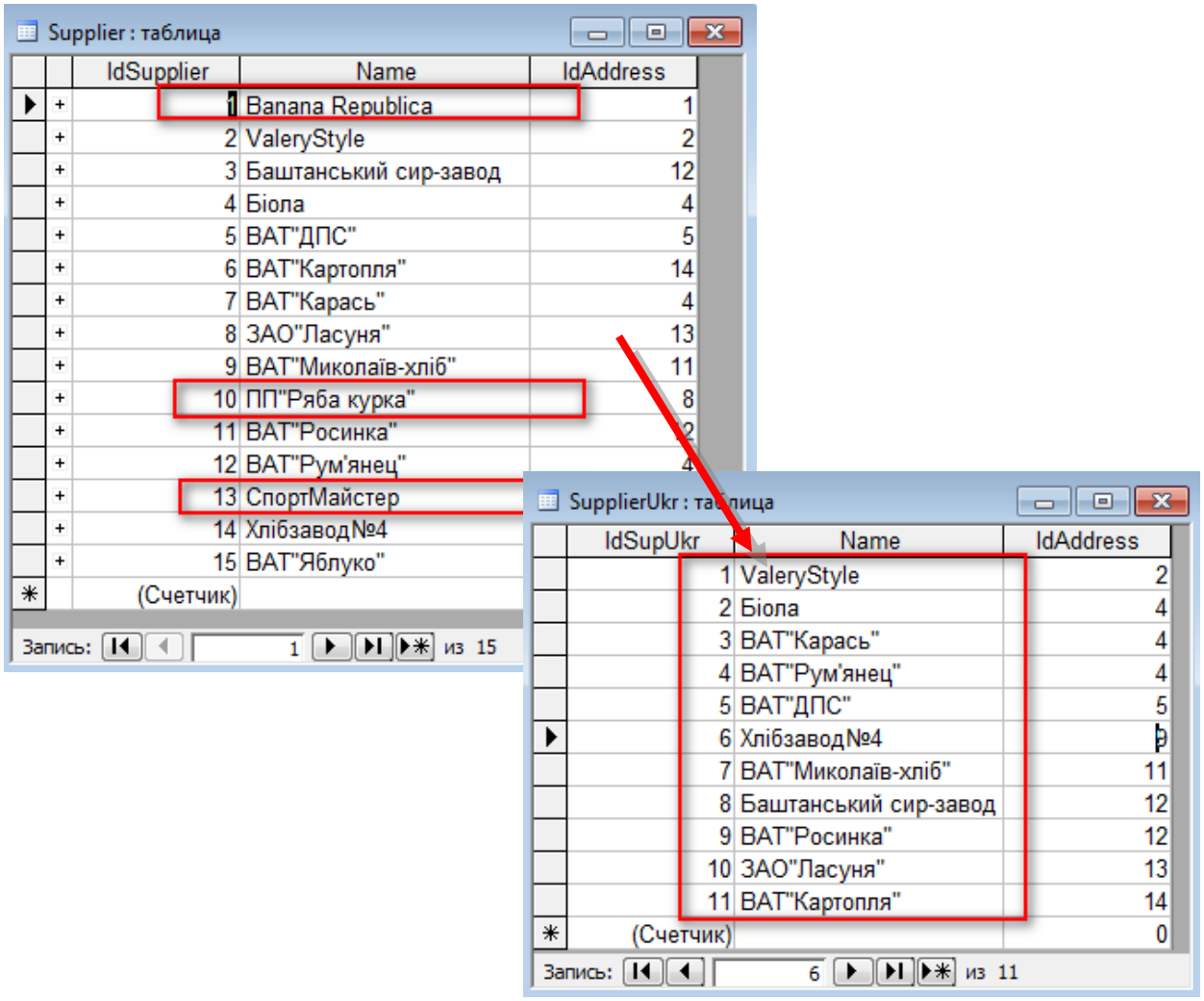


Рис. 1.108. Результат запиту з прикладу 1.108

Піктограма запитів на додавання даних буде відрізнятись від запитів на вибірку та створення нової таблиці та буде мати значок «+» із знаком оклику (рис. 1.109)

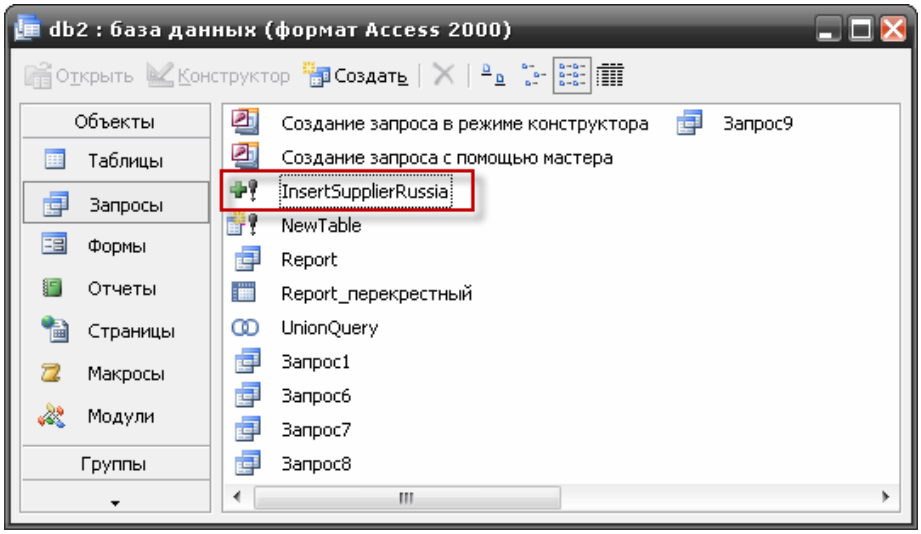


Рис. 1.109. Вікно об'єктів БД зі списком запитів. Запит на додавання

1.18.2. Оператор DELETE

За допомогою оператора `DELETE` з таблиці можна видалити довільну кількість полів. Дозволяється накладати умову, критерій на дані, що видаляються. Оскільки оператор `DELETE` видаляє запис повністю і дану операцію відмінити неможливо (лише через відновлення з резервної копії), слід перевіряти перед видаленням умову, щоб бути впевненим у вірності видалення даних.

При видаленні записів, існує одна особливість. Якщо видаляється запис з таблиці, яка входить у відношення 1:N (один-до-багатьох) і зі сторони 1 дозволене каскадне видалення, то записи, які відносяться до неї зі сторони N, також будуть видалені. Наприклад, якщо таке відношення вставлене для таблиць Customers і Orders, то видалення запису про клієнта автоматично викличе видалення запису про замовлення.

Синтаксис оператора **DELETE** наступний:

```
DELETE
FROM назва_таблиці
[WHERE умова];
```

Якщо умова не вказана, тоді будуть видалені всі записи вказаної таблиці, що відповідає її повному очищенню. Інакше видаленню будуть підлягати лише ті записи таблиці, які задовольняють критеріям.

Приклад 1.107. Видалити вміст таблиці Category, тобто видалити всі категорії товарів:

```
DELETE
FROM Category;
```

Після цього таблиця Category буде пустою.

Приклад 1.108. Видалити з бази даних виробника «MicroChips» Ltd.

```
DELETE
FROM Producer
WHERE Name='"MicroChips" Ltd.';
```

Приклад 1.109. Видалити всі товари, виробником яких є «MicroChips» Ltd. – без підзапиту не обійтись:

```
DELETE
FROM Product
WHERE IdProducer =
                (SELECT IdProducer
                 FROM Producer
                WHERE Name='"MicroChips" Ltd.');
```

Піктограма запиту на видалення також буде відрізнятись від інших запитів і буде мати вигляд хрестика із знаком оклику (рис. 1.110).

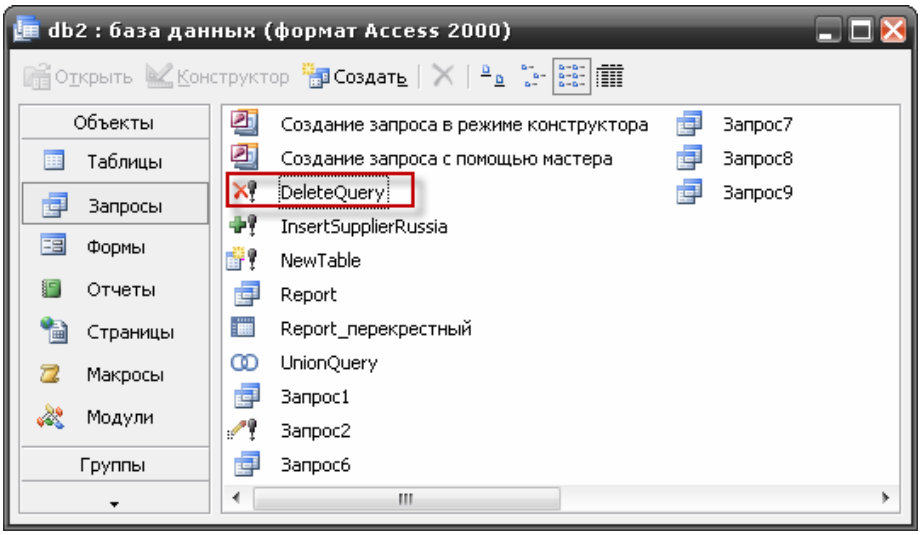


Рис. 1.110. Вікно об'єктів БД зі списком запитів. Запит на видалення

1.18.3. Оператор UPDATE

Оператор поновлення даних **UPDATE**. Він дозволяє змінити значення поля (-ів) у вказаній таблиці, як правило, згідно вказаному критерію. Даний оператор особливо корисний при зміні великої кількості записів або записів в кількох таблицях. Подібно оператору **DELETE**, операцію зміни неможна відмінити, тому перед її запуском потрібно впевнитись у правильності побудови критерію відбору.

Синтаксис оператора **UPDATE**:

```
UPDATE назва_таблиці
SET назва_поля = значення [,...]
[WHERE умова];
```

Інструкція **SET** визначає, які поля повинні бути змінені і на які саме значення. Аналогічно оператору **DELETE**, конструкція **WHERE** є необов'язковою і дозволяє вказати умову відбору записів, значення яких буде модифіковане. За замовчуванням змінюються всі дані вказаної таблиці.

Приклад 1.110. Замінити назву постачальника з кодом 3 на «Inter Ltd.»:

```
UPDATE Supplier
SET Name = 'Inter Ltd.'
WHERE IdSupplier = 3;
```

Приклад 1.111. встановити ціну поставки рівною 20 грн. для всіх товарів, що поставлялись постачальниками «ValeryStyle» та «СпортМайстер»:

```
UPDATE Delivery SET Price = 20
WHERE IdSupplier IN
                (SELECT IdSupplier
                 FROM Supplier
                 WHERE Name IN ('ValeryStyle',
                              'СпортМайстер'));
```

Змінити інформацію про поставку йогурта «Живинка», яка була здійснена 12/07/2013 наступним чином: ціну поставки збільшити на 10%, а кількість збільшити на 100 од.

Приклад 1.112.

```
UPDATE Delivery
SET Price = Price * 0.2,
    Quantity = Quantity + 100
WHERE DateDelivery = #12/07/2013# AND IdProduct IN
                                  (SELECT IdProduct
                                  FROM Product
                                  WHERE Name='Йогурт "Живинка"');
```

Піктограма запитів на поновлення набуде вигляду олівця із знаком оклику, відрізняючи його від інших запитів (рис. 1.111).

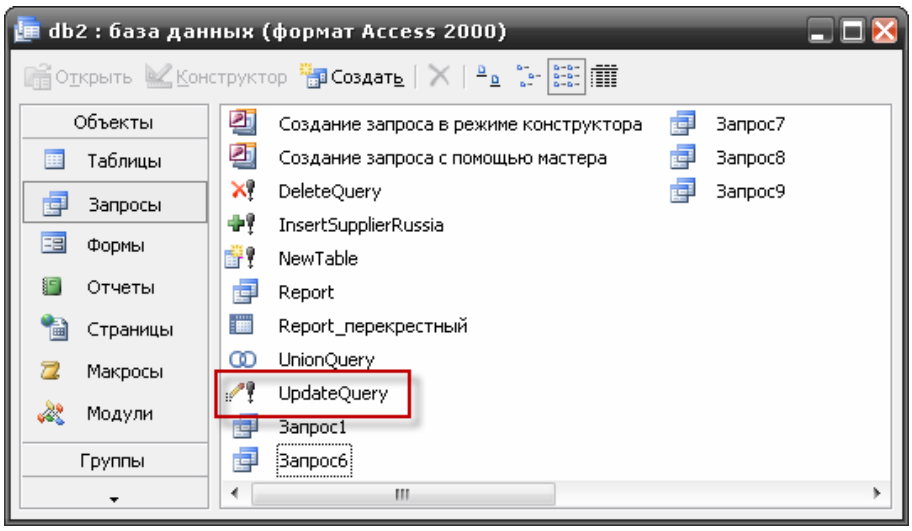


Рис. 1.111. Вікно об'єктів БД зі списком запитів. Запит на поновлення

1.19. Робота з уявленнями

Уявлення (view, огляд) – це запит, що зберігається та створюється на основі команди SELECT. Уявлення реально не утримає даних. Запит, який визначає уявлення, виконується тоді, коли до уявлення здійснюється звернення з іншого запиту, наприклад, SELECT, UPDATE та ін.

Призначення уявлень:

1. Зберження складних запитів.
2. Подання даних у вигляді, що зручний користувачу.
3. Зховування конфеденційної інформації.
4. Надання диференційованого доступу до даних.

Створення уявлень виконується командою **CREATE VIEW:**

```
CREATE [OR REPLACE] VIEW <ім'я_уявлення>
     [(<список_імен_стовпців>)]
  AS <запит> [WITH CHEC OPTION];
```

Запит (команда SELECT), на базі якого створюється уявлення, називають **визначальним запитом,** а таблиці, до яких здійснюється звернення у визначальному запиті – **базовими таблицями.** Визначальний запит за стандартом SQL не може включати речення ORDER BY.

Якщо не вказувати імена стовпців, то вони отримують назви за іменами, що перераховані у списку вибора визначального запиту. Вказувати імена стовпців обов'язково, якщо список вибор утримує функції агрегування або стовпці з однаковими іменами з різних таблиць.

Приклад 1.113. Створити уявлення «Продукт і виробник» (для зручного відображення виробників певних продуктів). Результат поданий на рис. 1.112.

```
CREATE VIEW prod_producer(Продукт, Виробник)
  AS SELECT e.Name, c.Name
     FROM Product e, Producer c
     WHERE e.IdProducer = c. IdProducer;

SELECT * FROM prod_producer;
```

	Продукт	Виробник
▶	Хліб білий	Хлібзавод№4
	Вода	Біола
	Хліб чорний	Хлібзавод№4
	Ватрушка	ВАТ"Миколаїв-хліб"
	Ковбаса"Рум'янець"	ПП"Ряба курка"
	Фарш"Добрий"	ПП"Ряба курка"
	Цукерки "Радість"	ЗАТ"Яблуко"
	Цукерки"Крабіки"	ВАТ"Рум'янец"
	Вода"Лімон"	ВАТ"Карась"
	Моршинська	ВАТ"Росинка"
	Міргородська	ВАТ"Росинка"
	Фарш"Добриня"	ПП"Ряба курка"
	Молоко"Корівка"	ВАТ"Рум'янец"
	Йогурт"Живинка"	ВАТ"Рум'янец"
	Сухарі"Житні"	Хлібзавод№4
	Банани	Banana Republica

Рис. 1.112. Результат запиту з приклада 1.113.

Приклад 1.114. Створити уявлення «Постачальники з Миколаєва» (для подання повного доступу до списку постачальників лише з Миколаєва місцевій маркетинговій групі користувачів). Результат поданий на рис. 1.113.

```
CREATE VIEW WSupplierMik
  AS SELECT Supplier.Name
     FROM Supplier, Address
     WHERE Supplier.IdAddress=Address.IdAddress
          AND Town='Миколаїв';
```

	Name
	ЗАО"Ласуня"
	ВАТ"Миколаїв-хліб"
▶	Хлібзавод№4

Рис. 1.113. Результат запиту з приклада 1.114.

Приклад 1.115. Створити уявлення «Постачальник» (без даних про адресу, для сховування конфеденційної інформації)

```
CREATE VIEW WSupplier
  AS SELECT IdSupplier, Name
     FROM Supplier;
```

Уявлення може бути **поновлювальним** або **не поновлювальним**. Поновлювальним є уявлення, при зверненні до якого можна поновити базову таблицю.

Приклад 1.116. Виконати поновлення базової таблиці Supplier через уявлення WSupplierMik.

```
UPDATE WSupplier
  SET Name=Name+'_WIP'
  WHERE Name Like'ЗАТ*';
```

Зміни будуть виконані в базовій таблиці та відобразяться в уявленні

```
SELECT * FROM WSupplier;
```

IdSupplier	Name
1	Banana Republica
2	ValeryStyle
3	Баштанський сир-завод
4	Біола
5	ВАТ"ДПС"
6	ВАТ"Картопля"
7	ВАТ"Карась"
8	ЗАТ"Ласуня"_WIP
9	ВАТ"Миколаїв-хліб"
10	ПП"Ряба курка"
11	ВАТ"Росинка"
12	ВАТ"Рум'янец"
13	СпортМайстер
14	Хлібзавод№4
15	ВАТ"Яблуко"

Рис. 1.114. Результат запиту з приклада 1.116.

Зміни, що вносяться, можуть вийти за межі визначального запиту і тому їх не можна буде побачити через уявлення. Якщо потрібно захистити дані від такого втручання, то потрібно в команді створення уявлення вказати ключові слова *WITH CHECK OPTION*: тоді система не прийме зміни, що виходять за межі визначального запиту.

За стандартом SQL-2 уявлення не є поновлювальним, якщо визначальний запит:

1. утримує ключове слово *DISTINCT*;
2. утримує множинні операції (*UNION та ін.*);
3. утримує речення *GROUP BY*;
4. посилається на інше непоновлювальне уявлення;
5. утримує обчислювальні вирази у списку вибору;
6. вибирає данні більш ніж з однієї таблиці.

1.20. Видалення об’єктів бази даних

Видалення об’єктів БД виконується за допомогою команди **DROP.**

DROP TABLE – видалення таблиці.

Синтаксис:

```
DROP TABLE <ім’я_таблиці> [ RESTRICT | CASCADE ];
```

Таблиця буде видалена без додаткового запиту на підтвердження разом з даними та деякими іншими об’єктами, існування котрих залежить від наявності таблиці (індекси, тригери та ін.). При застосуванні *CASCADE* разом з таблицею каскадно видаляються усі залежні від неї об’єкти (інші таблиці). Якщо вказати *RESTRICT*, то при наявності об’єктів, залежних від таблиці, що видаляється, операція буде відмінена.

DROP VIEW – видалення уявлення.

Синтаксис:

```
DROP VIEW <ім’я_уявлення>;
```

2. ЛАБОРАТОРНИЙ ПРАКТИКУМ

ЛАБОРАТОРНА РОБОТА № 1

Тема

ВВЕДЕННЯ В SQL. СТВОРЕННЯ ВІДНОШЕНЬ. ОПЕРАТОР SELECT.

Мета

Вивчити категорії команд SQL. Навчитись створювати таблиці та запити з використанням однієї таблиці за допомогою оператора SELECT.

Вимоги до звіту. За результатами роботи надати набір SQL-скриптів, який вирішує задачі з пункту «Самостійна робота»

ХІД РОБОТИ

1. Вивчити теоретичні відомості з пунктів 1.2–1.9.
2. Опрацювати практичні завдання з прикладів 1.19–1.55.
3. Виконати завдання до лабораторної роботи з п. «Самостійна робота».
4. Підготувати відповіді на контрольні питання.

САМОСТІЙНА РОБОТА

1. Створити фрагмент БД «Відділ кадрів», який складається з таблиць «Відділи» (Номер відділу (PK), Назва відділа), «Співробітники» (Номер відділа (FK), Табельний номер співробітника (PK), ПІБ співробітника, Посада, Оклад, Дата народження, Телефон, Дата прийняття на роботу), «Діти співробітників» (Табельний номер батька (FK), Ім’я дитини, Стать, Дата народження) за запропонованою структурою. Забезпечити зв’язки між таблицями.

Робота з БД «Магазин» (файл **Shop.mdb).**

2. Створити запит на вибірку всієї інформації про поставки товарів в магазин.
3. Вивести назви товарів, наявна кількість яких не вказана.
4. Створити запит на вибірку інформації про постачальників магазину, а саме їх назву та повну адресу. Адресу вивести в одному полі (відформатувати виведення).
5. Вивести інформацію про те, яких саме товарів було продано більше 10 упродовж від 01/12/2013 до 01/03/2014
6. Вивести назви товарів, що продавались упродовж останнього місяця (без повторень). *Примітка*! Для отримання інформації про поточний місяць слід скористатись необхідними функціями.
7. Вивести список товарів, що супроводжується назвами виробників, категорія яких «Бакалія».

8. Створити запит на вибірку інформації про *середню вартість продажу* на вказану в умові дату.
9. Вивести список всіх постачальників, із вказанням країн їх знаходження, назви яких містять літеру К, а місто розташування починається з літери М.
10. Вивести всі товари, що поставляються постачальниками «ПП Петров» і «ПП Іванов».
11. Вивести інформацію про всі товари, назви яких починаються з В по Л включно.
12. Вибрати всі товари з вказанням постачальника, ім'я виробника яких починається з К або М та категорія не «СокіВоди».
13. Вибрати всі товари із вказанням імені та країни їх виробника. *Умова*: країна виробника «Україна», «Росія» або «Польща» ціни поставки були < 50 грн., а дата поставки були в межах 01/01/2013 до сьогоднішнього дня.
14. Знайти товари, категорія яких «СокіВоди», кількість продажу яких більше 100. Вивести такі товари, їх виробників, постачальників та категорію.
15. Створити багатотабличний запит на вибірку інформації про поставку товарів в наступному вигляді: назва поставлених товарів, їх постачальники, категорії, дати поставки та загальну вартості їх поставки. Умова: лише трьох обраних постачальників. Відсортувати вибірку по назві товару в алфавітному порядку.
16. Створити багатотабличний запит на вибірку інформації про продаж товарів в наступному вигляді: назва проданих товарів, їх виробників, категорій, дати продажу, вартості продажу, з вказанням повної адреси виробників. Умова: виведена інформація не повинна стосуватись двох визначених виробників (наприклад, крім «ПП Іванов» і «ПП Петров»). Передбачити виведення повної адреси виробників в одному полі. Відсортувати вибірку за назвою товарів за зростанням та за їх вартістю за спаданням.

КОНТРОЛЬНІ ПИТАННЯ

1. Введення в SQL. Категорії команд SQL: DDL, DML, DCL
2. Типи запитів SQL
3. Створення таблиць: синтаксис, приклади.
4. Обмеження відношень.
5. Підготовка платформи для роботи з SQL в MS Access.
6. Запити з використанням однієї таблиці. Оператор SELECT
7. Вибірка з використанням оператора WHERE.
8. Сортування даних засобами мови SQL.
9. Багатотабличні запити на вибірку даних.
10. Декартова множина значень.

ЛАБОРАТОРНА РОБОТА № 2

Тема

ФУНКЦІЇ АГРЕГУВАННЯ. ГРУПУВАННЯ ДАНИХ.

Мета

Вивчити функції агрегування мовою SQL. Навчитись створювати запити з використанням групування даних. Оператори GROUP BY та HAVING.

Вимоги до звіту. За результатами роботи надати набір SQL-скриптів, який вирішує задачі з пункту «Самостійна робота»

ХІД РОБОТИ

1. Вивчити теоретичні відомості з пунктів 1.10–1.11.
2. Опрацювати практичні завдання з прикладів 1.56–1.64.
3. Виконати завдання до лабораторної роботи з п. «Самостійна робота».
4. Підготувати відповіді на контрольні питання.

САМОСТІЙНА РОБОТА

1. Вивести загальну кількість товарів, максимальну та середню ціну їх продажу.
2. Вивести на екран кількість нових товарів в магазині, тобто товарів, які були поставлені упродовж останнього тижня (7 днів).
3. Підрахувати загальну кількість товарів двох виробників, наприклад «ПП Іванов» і «ПП Петров».
4. Вивести інформацію про найменшу вартість поставки для кожного постачальника товарів за останній місяць (для отримання інформації про останній місяць скористайтесь стандартними функціями) в євро. Відсортувати вибірку за назвою постачальника за зростанням.
5. Підрахувати та вивести інформацію про загальну кількість продажу за кожен день в межах 01/01/2013 до поточного дня і вивести їх за спаданням кількості продажу.
6. Вивести назви товарів певного виробника, що поставлялись більш, ніж двома постачальниками.
7. Вивести інформацію про виробників, країни їх розташування, кількість товарів їх виробництва, які наявні в магазині. Умова: загальна вартість продажу товарів кожного виробника повинна бути в межах від 100 до 200 грн. Відсортувати вибірку по країні розташування за зростанням та кількості товарів виробників за спаданням.
8. Розрахувати та вивести кількість товарів, кожної категорії, які необхідно списати. До списання підлягають товари, які наявні в магазині і не продавались з дати їх поставки упродовж 3 місяців.

9. Вивести на екран кількість наявного товару за кожною категорією, при цьому враховувати лише товари, вартість поставки яких перевищувала 300 грн. Виведена інформація повинна стосуватись лише двох постачальників: ВАТ «Карась», ЗАТ «Яблуко».
10. Показати категорію, товарів якої в магазині знаходиться найменше.

КОНТРОЛЬНІ ПИТАННЯ

1. Функції агрегування. Необхідність використання.
2. Групування даних. Оператор GROUP BY.
3. Групування даних. Оператор HAVING

ЛАБОРАТОРНА РОБОТА № 3

Тема

ПІДЗАПИТИ. ОБ'ЄДНАННЯ ЗАПИТІВ ТА ТАБЛИЦЬ.

Мета

Навчитись створювати підзапити та виконувати об'єднання таблиць мовою SQL.

Вимоги до звіту. За результатами роботи надати набір SQL-скриптів, який вирішує задачі з пункту «Самостійна робота»

ХІД РОБОТИ

1. Вивчити теоретичні відомості з пунктів 1.12–1.14.
2. Опрацювати практичні завдання з прикладів 1.65–1.94.
3. Виконати завдання до лабораторної роботи з п. «Самостійна робота».
4. Підготувати відповіді на контрольні питання.

САМОСТІЙНА РОБОТА

1. Отримати інформацію про всіх виробників, товари яких продались більше, ніж 2 рази
2. Виявити набільш популярний товар в магазині, тобто той, який найбільше продавався.
3. Якщо загальну кількість товарів всіх категорій вважати за 100%, то необхідно підрахувати скільки товарів кожної категорії (у відсотковому відношенні) було продано.
4. Використовуючи підзапити вивести коди та назви всіх товарів, які поставляються лише визначеним постачальником, наприклад, ЗАТ «Яблуко».

5. Використовуючи підзапити вивести список товарів, їх ціни та категорії, які поставляються тільки одним постачальником.
6. Використовуючи підзапити вивести назви постачальників, які не поставляли вказаний товар (наприклад, «Йогурт»).
7. Використовуючи підзапити вивести на екран список виробників, які розташовані в тій же країні, що і, наприклад, постачальник «ПП Іванов».
8. Написати запит, який виводить на екран список постачальників однієї країни (наприклад, України). Використати для виведення результату самооб'єднання.
9. Підрахувати кількість постачальників, товари яких поставлялись в період з 01/03/2013 по 01/06/2014 і не були продані. Використайте для цього оператор EXISTS.
10. Виведіть список виробників, які розміщуються не в Україні. Відсортуйте вибірку за зростанням назв виробників. Для даного підзапиту не використовуйте об'єднання таблиць, а лише корельовані підзапити та оператор EXISTS.
11. Вивести на екран назву товару, постачальника, який його поставляв, його повну адресу (в одному полі), категорія яких «СокіВоди» та «Фрукти». Врахувати при виведенні лише ті товари, які поставляються приватними підприємцями.
12. Використовуючи підзапити знайти постачальників, товарів яких немає в продажу. Використайте для пошуку оператор ANY або SOME.
13. Відібрати всіх виробників, товарів яких в магазині в наявності більше, ніж будь-якого товару виробника ЗАТ «Ласуня».
14. Вивести інформацію про те, товарів яких виробників в базі даних не існує. Для виведення повноцінної інформації скористайтесь зовнішнім об'єднанням.
15. Відобразити всі товари, категорій «Кондитерські вироби» та «Хлібобулочні вироби» та назви постачальників, які їх поставляли (використати оператор UNION).
16. Отримати інформацію про кількість постачальників двох країн (наприклад, України і Росії), товари яких існують в базі даних. При цьому вивести окремо отриману інформацію та загальну суму всіх постачальників товарів. Скористайтесь для цього операторами UNION та UNION ALL.

Вимоги до виконання сам. роботи. Всі багатотабличні запити повинні бути написані згідно стандарту SQL2.

КОНТРОЛЬНІ ПИТАННЯ

1. Види та застосування вкладених підзапитів
2. Оператори EXIST, ANY, SOME, ALL
3. Оператори UNION та UNION ALL

4. Види об’єднань
5. Внутрішні об’єднання. Оператор INNER JOIN
6. Зовнішні об’єднання (OUTER JOIN) та його типи: ліве, праве та повне
7. Самооб’єднання таблиць

ЛАБОРАТОРНА РОБОТА № 4

Тема

ПЕРЕХРЕСНІ ЗАПИТИ. МОДИФІКАЦІЯ ДАНИХ ЗАСОБАМИ DML. РОБОТА З УЯВЛЕННЯМИ

Мета

Навчитись створювати перехресні запити та виконувати модифікацію даних засобами DML, працювати з уявленнями.

Вимоги до звіту. За результатами роботи надати набір SQL-скриптів, який вирішує задачі з пункту «Самостійна робота»

ХІД РОБОТИ

1. Опрацювати теоретичні відомості з пунктів 1.1, 1.15–1.20.
2. Виконати практичні завдання з прикладів 1.94–1.116.
3. Виконати завдання до лабораторної роботи з п. «Самостійна робота».
4. Підготувати відповіді на контрольні питання.

САМОСТІЙНА РОБОТА

Запити, модифікація даних засобами DML

1. Зробити запит на створення таблиці, що містить інформацію про 5 найбільш продаваємі товари.
2. Написати параметризований запит, який повертає 2 найдорожчі товари вказаної категорії.
3. Написати запит, який дозволить збільшити дату поставки кожного товару, яка відповідає шаблону на 2 дні. Шаблон на товар задається користувачем, тобто передається в якості параметра в запит.
4. Написати запит, який дозволяє переглянути всі товари певного виробника та необхідної категорії, при цьому дані про виробника і категорію вказуються при виклику
5. Засобами SQL зробити вставку нових даних в таблицю Product та Country.
6. Створити таблицю Delivery2014, яка має структуру, аналогічну таблиці Delivery. Дана таблиця повинна містити інформацію про всі поставки, які були здійснені упродовж 2014 року. З ціллю автоматизації процесу заповнення таблиці Delivery2014, написати необхідний запит на вставку.
7. Зменшити ціни на всі товари, які постачались двома постачальниками (наприклад, «Біола» та «Росинка») на 10% і тип націнки при цьому змінити на оптову.

8. Збільшити на 20% ціну продажу всіх товарів, виробник яких, наприклад, «Бондарчук» та які постачались певним постачальником (наприклад, «Горобець»).
9. Проставити дату поставки рівною сьогоднішній для всіх товарів, в яких така інформація відсутня.
10. Видалити з поточної бази даних всі товари, наявна кількість яких менше 100 од. (кг, шт. тощо), а ціна не перевищує 10 грн. за од.
11. Видалити всі товари кондитерської групи, продаж яких з початку року складає більше 100 кг.
12. Видалити всі товари, в яких ціна більша, ніж середня.
13. Засобами SQL створити місячний звіт, який містить інформацію про наступне: від яких постачальників виконана поставка товарів поточного року в розрізі категорій. *Примітка*! Прив'язки до конкретних дат бути не повинно, тобто поточний рік визначається програмно.
14. Засобами SQL створити річний звіт про середні ціни продажу товарів певної категорії. Категорія, за якою необхідно формувати звіт, задається користувачем кожен раз перед його створенням.

Уявлення

15. Створити уявлення «TOP 10 товарів, що є лідерами продаж» (Назва товару, Кількість продажів, Загальна сума продаж).
16. Створити уявлення «Постачальники певної країни» (Ім'я постачальника, Країна, Місто).

Реляційна алгебра (Завдання цього блоку виконуються на папері)

17. Нехай таблиця R(a, b, c, d, e) утримує n рядків, а таблиця S (f, g, h, i, j, k) – m рядків, та S.h – є зовнішнім ключем, який посилається на первинний ключ R.a.
    17.1. Написати вираз реляційної алгебри для декартова добутку відношень R та S. Скільки рядків буде у результуючій таблиці, яка отримана за допомогою цього виразу?
    17.2. Написати вираз реляційної алгебри для з'єднання відношень R та S по R.a= S.h. Яким може бути максимальне та мінімальне число рядків таблиці, що отримана в результатів виконання цієї операції? В якому випадку отримаємо максимальну кількість рядків, а в якому – мінімальну? Скільки стовпців буде в результуючій таблиці?
    17.3. Написати вираз реляційної алгебри для з'єднання відношень R та S по R.b= S.k. Яким може бути максимальне та мінімальне число рядків таблиці, що отримана в результатів виконання цієї операції? За яких умов отримаємо максимальну кількість рядків, мінімальну?
18. Користуючись відношеннями БД *«Магазин»* описати кожне з наведених нижче реляційних виразів та відношень, які будуть отримані в результаті їх виконання.
    18.1. $\pi_{Name,\ Quantity}\ (\sigma_{Price>110}\ (Product))$

18.2. $\pi_{IdProduct,\ IdDelivery}\ (\sigma_{DateDelivery\ ='10/09/2014'}\ (Delivery))$

18.3. $\pi_{Name,\ Price,\ DateSale}\ (\sigma_{IdProduct\ ='18'}\ (Product\ ><_{IdProduct}\ Sale))$

18.4. $\pi_{Name,Price,Quantity,IdDelivery}\ (\sigma_{DateDelivery\ >'01/09/2012'}\ (Product\ ><_{IdProduct}\ Delivery))$

КОНТРОЛЬНІ ПИТАННЯ

1. Перехресні запити
2. Запити з параметрами
3. Запити на створення таблиці. Конструкція TOP
4. Модифікація даних засобами DML. Оператор INSERT
5. Модифікація даних засобами DML. Оператор DELETE
6. Модифікація даних засобами DML. Оператор UPDATE
7. Основні операції реляційної алгебри. Приклади.
8. Спеціальні реляційні операції. Приклади.

3. ДОДАТКОВІ ЗАВДАННЯ ДЛЯ ІНДИВІДУАЛЬНОГО ВИКОНАННЯ

Завдання 1. Нехай БД утримує наступні три таблиці:

- `Студенти (ID_СТУДЕНТА, Ім'я, Телефон, Адреса, ID_курса)` – данні про студентів, в тому числі і про курси, які вони вивчають;
- `Курси (ID_курса, Назва)` – данні про курси;
- `Викладачі (ID_викладача, Ім'я, Телефон, Адреса, ID_курса)` – данні про викладачів, в тому числі і про курси, заняття по яким вони проводять.
- В таблиці `Студенти` кожен студент може бути записаний на декількох курсах. Аналогічно, один і той же викладач може проводити заняття за декількома курсами. Тобто, `стовпці ID_курса` в цих таблицях не повинні містити унікальні значення. В таблиці Курси, напроти, стовбець `ID_курса` утримує тільки унікальні значення.

Сформулюйте SQL-вираз для створення вказаних таблиць так, щоб таблиці `Студенти` та `Викладачі` буди пов'язані з таблицею Курси.

Завдання 2. Для БД із завдання 1 сформулювати запит, який повертає таблицю, що утримує:

а) найменування курсів та імена викладачів, які проводять заняття по даним курсам;

б) найменування курсів та імена студентів, які вивчають данні курси;

в) імена студентів та імена викладачів, які проводять заняття з даними студентами.

Завдання 3. Для БД із завдання 1 сформулювати запит, який повертає таблицю, яка містить найменування курсів, викладачів та студентів. Кожен рядок у таблиці повинен містити найменування деякого курсу, ім'я викладача, який проводить заняття за даним курсом, та ім'я студента, який його вивчає.

Завдання 4. Додати до БД із завдання 1 таблицю `Контакти (Ім'я, Адреса, Телефон, Примітка)`. Сформулюйте запит, який додає до неї відповідні дані із таблиць `Студенти` та `Викладачі`. Розв'язати дане завдання, записуючи при цьому у стовбець `Примітка` слово `'Студент'` або `'Викладач'`, в залежності від того, з якої таблиці отримані дані для додавання в таблицю `Контакти`.

Завдання 5. Створити **БД НДР (**науково-дослідна робота**)**, в якій зберігається інформація про виконані виплати спеціалістам за виконану роботу за певними етапами НДР, вказавши потрібні обмеження для полів та ключові поля.

БД складається з трьох таблиць:

```
R1 = (ПІБ, Відділ);
R2 = (Відділ, Етап);
R3 = (ПІБ, Етап, Нарахування).
```

Завдання 6. У БД НДР із завдання 5 додати до таблиці R1 дані:

	R1
ПІБ	**Відділ**
Сєменів Т.Т.	03
Прасов С.М.	03
Мехова И.И.	03
Чемсов Я.Ю.	04
Яров И.М.	04

Додати до таблиці R2 дані:

	R2
Відділ	**Етап**
03	Етап_1
03	Етап_2
03	Етап_3
04	Етап_3
04	Етап_4

Додати до таблиці R3 дані:

		R3
ПІБ	**Етап**	**Нарахування (грн)**
Сєменів Т.Т.	Етап_1	1000
Прасов С.М.	Етап_1	2000
Мехова И.И.	Етап_1	500
Сєменів Т.Т.	Етап_2	500
Прасов С.М.	Етап_2	500
Мехова И.И.	Етап_2	1000
Прасов С.М.	Етап_3	1000
Мехова И.И.	Етап_3	1000
Чемсов Я.Ю.	Етап_3	2000
Чемсов Я.Ю.	Етап_4	2000
Яров И.М.	Етап_4	3000

Завдання 7-10 виконуються для БД НДР

Завдання 7. Для БД НДР із завдання 6.

а) Додати до таблиці `ТИМЧАСОВА`, що має стовпці `ПІБ, Етап`, наявні в БД відомості про прізвища спеціалістів, а також про етапи, за якими відбулося нарахування зарплати.

б) Перейменувати відділ 03 на 03_1.

в) Збільшити всі нарахуванням на `Етапі 1` на 10%.

г) Видалити відомості про `Етапі 4`.

д) Видалити всі рядки з реляційного відношення `R1`. Запит скласти на папері.

Завдання 8. Для кожного спеціаліста визначити суму, яка виплачена за роботу за даною темою, та кількість зроблених йому виплат.

Завдання 9. В умовах попереднього запиту вивести інформацію, яка стосується тільки тих спеціалістів, яким здійснювалося нарахування більш одного разу.

Завдання 10. Вивести список співробітників відділу 03, які приймали участь у виконанні Етапа_3.

Завдання 11. Створити **БД Сесія**, в якій зберігається інформація про екзаменаційні оцінки студентів, отриманими ними в сесію за певними дисциплінами. Вказати потрібні обмеження для полів та ключові поля.

БД складається з трьох таблиць:

S1 = (ПІБ, Дисципліна, Оцінка) – утримує відомості про результати сесії;

S2 = (ПІБ, Група) – відомості про склад групи;

S3 = (Група, Дисципліна) – перелік іспитів, які потрібно здавати.

Завдання 12. У БД Сесія із завдання 11 додати до таблиці S1 дані:

S1

ПІБ	Дисциплина	Оцінка
Мур С.М.	Бази даних	4
ЦукановТ.Т.	Бази даних	5
Думска М.Т.	Бази даних	3
Дрозд Г.Р.	Бази даних	4
Мур С.М.	Історія	4
ЦукановТ.Т.	Історія	5
Думска М.Т.	Історія	3
ЦукановТ.Т.	Математика	5
Думска М.Т.	Математика	4
Дрозд Г.Р.	Математика	5
Петрова С.О.	Програмування	5
Часов І.І.	Інформатика	4
Іванова Я.С.	Інформатика	5
Кріс Р.О.	Інформатика	3
Часов І.І.	Аналіз даних	5
Іванова Я.С.	Аналіз даних	4
Часов І.І.	Програмування	4
Іванова Я.С.	Програмування	4
Кріс Р.О.	Програмування	5
Фірсова Л.Р.	Програмування	3

Додати до таблиці S2 данні:

S2

ПІБ	Група	Куратор
Мур С.М.	562	

ЦукановТ.Т.	562	
Думска М.Т.	562	
Дрозд Г.Р.	562	
Петрова С.О.	562	
Часов І.І.	562	
Іванова Я.С.	562	
Кріс Р.О.	562	
Фірсова Л.Р.	562	

Додати до таблиці S3 дані:

S3

Група	Дисципліна
362	Бази даних
362	Історія
362	Математика
362	Програмування
362	Інформатика
362	Аналіз даних

Завдання 13-16 виконуються для БД Сесія

Завдання 13. Додати до таблиці ТИМЧАСОВА, що має стовпці Назва_групи, Дисципліна, Оцінка, наявні в БД відомості про номер групи, а також про дисципліни та оцінки групи.

а) Встановити куратором 362 групи Іванова В.С.

б) Зменшити всім студентам, які здавали дисципліну «Бази даних», оцінку на 1 бал.

в) Видалити всі рядки з реляційного відношення S3.

г) Видалити з таблиці S1 ті предмети, з яких отримана 2.

Завдання 14. Для кожної дисципліни визначити кількість студентів, як склали іспит.

Завдання 15. В умовах запиту із завдання 14 вивести інформацію, яка стосується тільки тих дисциплін, за якими кількість складених іспитів перевищує три.

Завдання 16. Вивести групи, в яких по одній дисципліні на іспити отримано більше однієї п’ятірки.

Завдання 17. Нехай задано реляційну схему БД **ВНЗ**

```
ФАКУЛЬТЕТ (#F, Назва, Декан, Корпус, Фонд)
КАФЕДРА (#D, #F, Назва, #ЗАВІДУВАЧ, Корпус, Фонд)
ВИКЛАДАЧ (#T, #D, Прізвище, Посада, Телефон)
ГРУПА (#G, #D, Курс, Номер, Кількість, #КУРАТОР)
ПРЕДМЕТ (#S, Назва)
АУДИТОРІЯ (#R, Номер, Корпус, Місткість)
ЛЕКЦІЯ (#T, #G, #S, #R, Тип, День, Тиждень)
```

На рис.3.1 умовно зображено зв’язки між відношеннями описаної схеми.

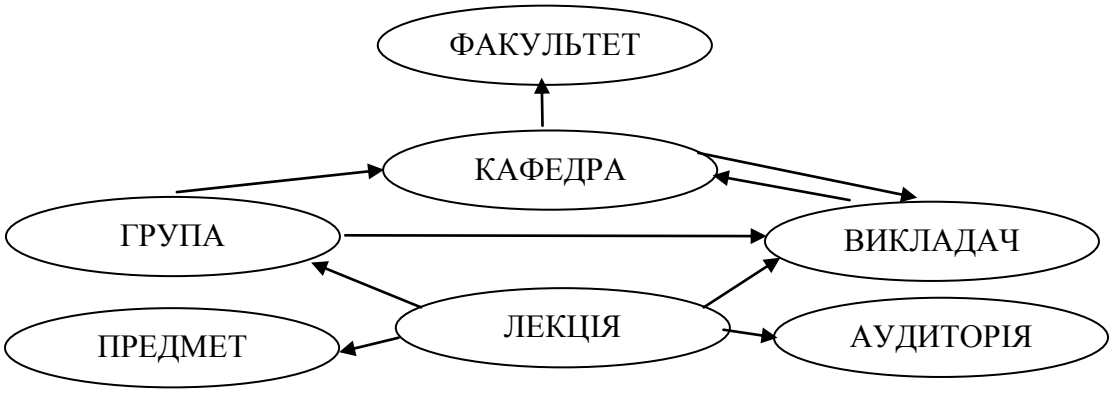


Рис. 3.1. Схематичне подання взаємозв’язків між таблицями БД ВНЗ

а) Створити зазначену вище БД ВНЗ.

Наприклад, для таблиці ФАКУЛЬТЕТ задати:

#F	Назва	Декан	Корпус	Фонд
NUMBER (10)	CHAR (10)	CHAR (25)	CHAR (5)	NUMBER (6.2)

б) Додати до таблиці ФАКУЛЬТЕТ рядок з даними.

#F	Назва	Декан	Корпус	Фонд
15	Механіко-математичний	Сидоров С.С.	2	25000

Додати ще 5 рядків на власний розсуд.

в) Додати до таблиці КАФЕДРА рядок з даними

#D	#F	Назва	ЗАВІДУВАЧ	Корпус	Фонд
03	15	ПМ та ІКТ	Петренко	5	5500

Додати ще 5 рядків на власний розсуд.

г) Додати до таблиці ВИКЛАДАЧ рядок з даними:

#T	#D	Прізвище	Посада	Телефон
173	13	Рєзніченко	NULL	526-18-15

Додати ще 5 рядків на власний розсуд.

д) Додати до таблиці ТИМЧАСОВА, що має стовпці Назва_факультету.

е) Назва_кафедри, Прізвище_викладача, наявні в БД відомості про викладачів, а також про кафедри та факультети, де вони працюють.

ж) Встановити фонд механіко-математичного факультету рівним 250 300.

з) Встановити фонд усіх факультетів рівним 260 500.

и) Збільшити всім факультетам фонд фінансування на 10%.

к) Збільшити всім кафедрам механіко-математичного факультету фонд фінансування на 5%.

л) Видалити відомості про лекції, що читаються по суботах та неділях.

м) Видалити всі рядки з реляційного відношення ПРЕДМЕТ.

н) Видалити з таблиці ПРЕДМЕТ ті предмети, з яких не читається лекцій.

Завдання 18-41 виконуються для БД ВНЗ.

Завдання 18. Визначити групи, де кількість студентів становить від 40 до 50 осіб, а також курси, які їм відповідають.

Завдання 19. Визначити прізвища та посади професорів та асистентів

Завдання 21. Визначити всі кафедри механіко-математичного факультету.

Завдання 22. Визначити усіх викладачів кафедри ПМ та ІКТ та їх телефонні номери.

Завдання 23. Вивести список усіх викладачів механіко-математичного факультету разом із їх телефонними номерами.

Завдання 24. Визначити номери груп першого курсу кафедри Математики.

Завдання 25. Визначити номери груп першого курсу кафедри Математики та прізвища їхніх кураторів.

Завдання 26. Визначити лекції, на яких кількість студентів у групі перевищує кількість місць в аудиторії. Вивести номери аудиторій та груп, назви дисциплін, що викладаються, а також дні й тижні проведення лекцій.

Завдання 27. Визначити кількість кафедр на механіко-математичному факультеті.

Завдання 28. Визначити кількість предметів, що викладається.

Завдання 29. Визначити місткість усіх аудиторій у корпусі 2.

Завдання 30. Визначити кількість студентів механіко-математичного факультету.

Завдання 31. Визначити найбільший фонд із фінансування серед кафедр механіко-математичного факультету.

Завдання 32. Визначити місткість усіх аудиторій корпусу 2, кількість місць у найменшій та найбільшій аудиторіях, середню місткість.

Завдання 33. Визначити кількість студентів на кожній кафедрі механіко-математичного факультету.

Завдання 34. Визначити кількість викладачів на кожному факультеті.

Завдання 35. Для кожного викладача визначити кількість дисциплін, які він викладає.

Завдання 36. Отримати фонди факультетів з таблиці `ФАКУЛЬТЕТ`, а також обчислити їх як суми фондів кафедр.

Завдання 37. Визначити факультети, де власний фонд перевищує сумарний фонд усіх кафедр.

Завдання 38. Визначити факультети, в яких власний фонд перевищує сумарний фонд усіх кафедр на 2000.

Завдання 39. Вивести прізвища викладачів за спаданням.

Завдання 40. Вивести за зростанням кількості викладачів на всіх кафедрах навчального закладу.

Завдання 41. Підрахувати середню величину фонду кафедри за умови, що фонд задано.

4. ТЕСТОВІ ПИТАННЯ ДЛЯ САМОКОНТРОЛЮ

1. **Хто вперше визначив реляційну алгебру**:
    a. Р. Бойс
    b. Е. Кодд
    c. Р. Фейгін
    d. К. Дейт

2. **Обрати традиційні операції з множинами, модифіковані для таблиць (відношень)**:
    a. Об’єднання
    b. Різниця
    c. Добуток
    d. Перетин
    e. Ділення
    f. З’єднання

3. **Які реляційні операції є спеціальними**:
    a. Множення
    b. Вибірка
    c. Проекція
    d. З’єднання
    e. Розмежування
    f. Ділення

4. **Що з перерахованого є операціями поновлення**:
    a. Видалення запису
    b. Переміщення запису
    c. Вставка запису
    d. Каскадування запису
    e. Поновлення запису

5. **Що буде результатом операції об’єднання**:
    a. Об’єдання кортежів двох таблиць
    b. Об’єдання атрибутів таблиць за деяким полем
    c. Усі кортежі першого відношення, які також входять до другого відношення
    d. Нічого з вищеперерахованого

6. **Що буде результатом операції перетину**:
    a. Усі кортежі першого відношення, які також входять у друге відношення
    b. Об’єднання кортежів двох таблиць
    c. Об’єднання атрибутів таблиць за деяким полем
    d. Нічого з вищеперерахованого

7. **Що буде результатом операції ділення**:
    a. Об’єднання атрибутів таблиць за деяким полем
    b. Об’єдання кортежів двох таблиць
    c. Усі кортежі першого відношення, які також входять у друге відношеня
    d. Нічого з вищеперерахованого

8. **Які з наступних відношень є правильними**:
    a. Операція проекції фактично є вибором деяких атрибутів з вихідного відношення
    b. Операція об’єднання може бути проведена над таблицями з різною кількістю атрибутів
    c. Результатом операції вибірки будуть усі кортежі вихідного відношення, що відповідають заданій умові
    d. Операція з’єднання може бути проведена над таблицями з різною кількістю атрибутів

9. **Що таке DML**:
    a. Data Manipulation Language
    b. Data Macro Language
    c. Data Meta-Language
    d. Direct Macro Language

10.Які з наступних розділів можуть зустрічатися в операторі SELECT:
    a. SORT
    b. GROUP BY
    c. REVERSE
    d. HAVING
    e. COMPUTE

11.**Що означає ключове слово ALL в операторі** SELECT:

a. З таблиці будуть вибрані усі записи, навіть ті, які не задовольняють умові в розділі WHERE
b. Ключове слово ALL більш не застосовується в операторі SELECT
c. З таблиці будуть вибрані усі записи, в тому числі і однакові

12.**Для чого застосовується ключове слово AS в операторі SELECT**:

a. Для порівняння атрибутів
b. Для повернення усіх записів
c. Для перейменування атрибута
d. Для отримання набору унікальних записів, які не повторюються

13.Яке ключове слово потрібно вказати для отримання набору унікальних записів, які не повторюються:

a. DISTINCT
b. PERCENT
c. ALL
d. TOP

14.Чи можливо додати до імені псевдоніму недопустимі символи:

a. Ні, ім’я псевдоніму повинно задовольняти стандартним правилам іменування об’єктів
b. Так, якщо ім’я псевдоніму помістити в круглі дужки
c. Так, якщо ім’я псевдоніму помістити в квадратні дужки
d. Так, якщо в списку вибірки вказати зірочку

15.Умова на групи під час вибірки з групуванням задається за допомогою оператора:

a. HAVING
b. WHERE
c. BETWEEN
d. LIKE

16.**Для пошука значень за шаблоном використовується оператор**:

a. IN
b. LIKE
c. IS
d. TEMPLATE

17.Що з нижчеперерахованого не є функцією агрегації:

a. COUNT
b. AVG
c. SUM
d. INTO

18.**Оператор об’єднання таблиць це**:

a. GROUP BY
b. SELECT
c. JOIN
d. BETWEEN

19.Які з перерахованих нижче тверджень є правильними для оператора SELECT:

a. Можна використовувати символ * для вибірки усіх полів
b. Ключове слово TOP призначене для виводу перших *n* записів
c. Розділи FROM, WHERE, GROUP BY, HAVING можна використовувати в довільному порядку
d. За замовчуванням SELECT виводить набір записів, який відсортований за першим атрибутом

20.**Які види з’єднань декількох таблиць існують**:

a. Вибіркове
b. Внутрішнє
c. Перехресне
d. Послідовне
e. Зовнішнє

21.Який оператор використовується для об’єднання декількох наборів результатів:

a. JOIN
b. SELECT
c. UNION

22.Яка агрегатна функція повертає кількість значень у списку, відмінних від NULL:

a. Sum ([all | distinct] вираз)
b. Count ([all | distinct] вираз |)
c. Avg ([all | distinct] вираз)
d. Max ([all | distinct] вираз)

23.Підзапити бувають:

a. Внутрішні
b. Зовнішні
c. Вкладені
d. Зв’язані
e. Об’єднанні

24.Під час зміни даних в таблиці за допомогою запиту використовується оператор:

a. EDIT
b. UPDATE
c. INSERT
d. CHANGE

25.Який розділ застосовується для групування записів за полями або виразами:

a. SORT BY
b. GROUP BY
c. GROUP ON

26. Нехай дана таблиця Автор з полями

```
Прізвище CHAR (50),
Стать CHAR(3),
Дата_народження DATETIME,
Телефон CHAR(9),
Кількість_праць INT,
Місто CHAR(15);
```

Знайти авторів, кількість праць яких перевищує 10, але не більше 20.

a. SELECT Прізвище, Кількість_праць
   FROM Автор
   WHERE Кількість_праць >10 And Кількість_праць <=20

b. SELECT Прізвище, Кількість_праць
   FROM Автор
   WHERE Кількість_праць Between 10 And 20

c. SELECT Прізвище, Кількість_праць
   FROM Автор
   WHERE Кількість_праць Between 11 And 20

d. SELECT Прізвище, Кількість_праць
   FROM Автор
   WHERE Кількість_праць >=11 Or Кількість_праць <=19

27. Нехай дана таблиця Автор (див. Питання 25). Знайти місто, в яких мешкають молоді автори (до 25 років), які видали більш ніж 10 книг.

a. SELECT Місто
   FROM Автор
   WHERE Дата_народження>’1/1/1980’ AND Кількість_праць >10

b. SELECT Місто, Дата_народження, Кількість_праць
FROM Автор
WHERE Кількість_праць >25 AND Кількість_праць >10

c. SELECT Місто
FROM Автор
WHERE Дата_ народження >25 OR Кількість_праць >10

d. SELECT DISTINCT Місто
FROM Автор
WHERE Дата_народження>’1/1/1980’ OR Кількість_праць >10

28. Нехай дана таблиця Автор (див. Питання 26). Визначити авторів, прізвища яких починаються на літери «Б» або «Г» та мають слог «-ор-».

a. SELECT Прізвище
FROM Автор
WHERE (Прізвище Like ”Б%” Or Прізвище Like ”Г%”) And Прізвище Like ”%ор%”

b. SELECT Прізвище
FROM Автор
WHERE Прізвище Like ”Б%” Or Прізвище Like ”Г%” And Прізвище Like ”%ор%”

c. SELECT Прізвище
FROM Автор
WHERE (Прізвище Like ”Б%” AND Прізвище Like ”Г__) And Прізвище Like ”_ор_”

d. SELECT Прізвище
FROM Автор
WHERE (Прізвище Like ”Б_%” Or Прізвище Like ”Г%_”) Or Прізвище Like ”%ор%”

29. Нехай дана таблиця Автор (див. Питання 26). Вивести за алфавітом прізвища авторів з Миколаєва, в телефонному номері котрих на другому та третьому місцях стоять цифри 5 або 8, а останніми є цифри 34.

a. SELECT Прізвище, Місто, Телефон
FROM Автор
WHERE Місто =”Миколаїв” AND (Телефон Like ”_[5,8]%”
Or Телефон Like ”__[5,8]%”) And Телефон Like ”%[3][4]”
ORDER BY Прізвище

b. SELECT Прізвище, Місто, Телефон
FROM Автор
WHERE Місто =” Миколаїв ” AND Телефон Like ”_[5,8]%”
Or Телефон Like ”__[5,8]%” And Телефон Like ”%[3][4]”
ORDER BY Прізвище

c. SELECT Прізвище, Місто, Телефон
FROM Автор
WHERE Місто =” Миколаїв ” AND (Телефон Like ”_[5,8]%”
AND Телефон Like ”__[5,8]%”) And Телефон Like ”%[3][4]”
ORDER BY Прізвище

d. SELECT Прізвище, Місто, Телефон
FROM Автор
WHERE Місто =” Миколаїв ” OR (Телефон Like ”_[5,8]%”
AND Телефон Like ”__[5,8]%”) OR Телефон Like ”%[3][4]”
ORDER BY Прізвище

30. Дано таблиці `Рейс та Квиток`

```
CREATE TABLE Рейс
(Номер_рейса                INT,
 Кінцевий_пункт             CHAR(30),
 Дата_виліту                DATETIME,
 Тривалість_маршрута        INT,
 Кількість_квитків          INT,
 Вартість                   CURRENCY)
```

```
CREATE TABLE Квиток
(Номер_місця                INT,
Номер_рейса                 INT,
Дата_продажу                DATETIME,
Вартість                    CURRENCY,
Прізвище пасажира           CHAR(20))
```

Розрахувати загальну вартість білетів для кожного рейсу, 5% податок з продажів і загальних прибуток рейса.

a) SELECT Номер_рейса, Вартість *Кількість_квитків AS Заг_вартість, Вартість * Кількість_квитків 0.05 AS Податок, Вартість * Кількість_квитків 0.95 AS Доход

```
FROM Рейс
```

b)
```
SELECT Номер_рейса,
   Вартість * Кількість_квитків AS Заг_вартість,
   Заг_вартість * 0.05 AS Податок,
   Заг_вартість *0.95 AS Дохід
FROM Рейс
```

c)
```
SELECT Номер_рейса,
  Заг_вартість = Вартість * Кількість_квитків,
  Податок =Общ_стоимость* 0.05,
  Дохід =Общ_стоимость *0.95
 FROM Рейс
```

d)
```
SELECT Номер_рейса,
   @s= Вартість * Кількість_квитків,
   @n= Вартість * Кількість_квитків 0.05,
   @d= Вартість * Кількість_квитків 0.95
FROM Рейс
```

31. **Для таблиць із завдання 30 визначити кількість проданих на кожен рейс квитків.**

a)
```
SELECT Рейс.Номер_рейса,
Count(Билет.Прізвище_пасажира) AS Кіл_пасажирів
FROM Рейс INNER JOIN Квиток
            ON Рейс.Номер_рейса = Квиток.Номер_рейса
GROUP BY Рейс.Номер_рейса
```

b)
```
SELECT Рейс.Номер_рейса,
     Кол_пассажиров= Count(Билет. Прізвище_пасажира)
FROM Рейс INNER JOIN Квиток
              ON Рейс.Номер_рейса =Билет.Номер_рейса
GROUP BY Рейс.Номер_рейса
```

c)
```
SELECT Рейс.Номер_рейса,
    Count(Квиток. Прізвище_пасажира) AS Кол_пассажиров
FROM Рейс INNER JOIN Квиток
              ON Рейс.Номер_рейса = Квиток.Номер_рейса
```

d)
```
SELECT Номер_рейса,
     Count(Прізвище_пасажира) AS Кіл_ пасажирів
FROM Рейс INNER JOIN Квиток
              ON Рейс.Номер_рейса = Квиток.Номер_рейса
GROUP BY Номер_рейса
```

32. Для таблиць з питання 30 визначити на яку суму були продані білети в день виліту?

a) SELECT Sum(Квиток.Стоимость) AS Общ_Стоимость,
Квиток.Дата_продажу
FROM Рейс INNER JOIN Квиток
ON Рейс.Номер_рейса = Квиток.Номер_рейса
WHERE Квиток.Дата_продажу=Рейс.Дата_виліту
GROUP BY Квиток.Дата_продажу

b) SELECT Sum(Квиток.Вартість) AS Заг_вартість,
Квиток.Дата_продажу
FROM Рейс INNER JOIN Квиток
ON Рейс.Номер_рейса = Квиток.Номер_рейса
GROUP BY Квиток.Дата_продажу
HAVING Квиток.Дата_продажу=Рейс.Дата_виліту

c) SELECT Sum(Квиток.Вартість) AS Заг_Вартість
FROM Рейс INNER JOIN Квиток
ON Рейс.Номер_рейса = Квиток.Номер_рейса
WHERE Квиток.Дата_продажу=Рейс.Дата_виліту
GROUP BY Квиток.Дата_продажу

d) SELECT Квиток.Дата_продажу,
Заг_Вртість=Sum(Квиток.Стоимость)
FROM Рейс INNER JOIN Квиток
ON Рейс.Номер_рейса =Билет.Номер_рейса
WHERE Квиток.Дата_продажу=Рейс.Дата_виліту
GROUP BY Квиток.Дата_продажу

33.**Для таблиць з питання 30 вивести список міст, куди було продано більш 200 квитків на суму, що перевищує 1000 грн.**

a) SELECT Рейс.Кінцевий_пункт,
Count(Квиток.Прізвище_пасажира) AS Кіл_пасажирів,
Sum(Рейс. Вартість) AS Заг_ Вартість
FROM Квиток INNER JOIN Рейс
ON Квиток.Номер_рейса =Рейс.Номер_рейса
GROUP BY Рейс. Кінцевий_пункт
HAVING Count(Квиток. Прізвище_пасажира)>=200 AND
Sum(Рейс. Вартість)>10000

b) SELECT Рейс. Кінцевий_пункт,
Count(Квиток. Прізвище_пасажира) AS Кіл_пасажирів,
Sum(Рейс. Вартість) AS Заг_ Вартість
FROM Квиток INNER JOIN Рейс
ON Квиток.Номер_рейса =Рейс.Номер_рейса
GROUP BY Рейс.Кінцевий_пункт
HAVING Кіл_пасажирів>=200 AND Общ_ Вартість >10000

c) SELECT Count(Квиток. Прізвище_пасажира),
Sum(Рейс. Вартість)
FROM Квиток INNER JOIN Рейс
ON Билет.Номер_рейса =Рейс.Номер_рейса
GROUP BY Рейс. Кінцевий_пункт
HAVING Count(Квиток. Прізвище_пасажира)>=200
AND Sum(Рейс. Вартість)>10000

d) SELECT Рейс. Кінцевий_пункт,
Count(Квиток. Прізвище_пасажира) AS Кол_пассажиров,
Sum(Рейс. Вартість) AS Заг_Вартість
FROM Квиток INNER JOIN Рейс
ON Квиток.Номер_рейса =Рейс.Номер_рейса
HAVING Count(Квиток. Прізвище_пасажира)>=200
AND Sum(Рейс.Вартість)>10000

4. ВАРІАНТ КОНТРОЛЬНОЇ РОБОТИ «Робота з запитами SQL»

В якості навчальної БД для завдань №1 та №2 використовується база «Борей» (див. Додаток В).

Завдання 1

Чи мають тексти запитів помилку? Якщо мають, то в чому вона полягає? Якщо запит правильний, що буде результатом виконання запиту?

Завдання 1.1.

```
SELECT Клиенты.Название, Заказы.КодЗаказа,
       Доставка.Название, Заказы.ДатаРазмещения,
       Заказы.ДатаРазмещения
FROM Клиенты LEFT JOIN (Доставка RIGHT JOIN Заказы
            ON Доставка.КодДоставки = Заказы.Доставка)
         ON Клиенты.КодКлиента = Заказы.КодКлиента
ORDER BY Клиенты. Название, Заказы.КодЗаказа;
```

Завдання 1.2.

```
SELECT Типы.Категория,
       Count(Товары.КодТовара) AS [Count-КодТовара],
       Max(Товары.Цена) AS [Max-Цена],
       Типы.Описание,
       Min(Товары.МинимальныйЗапас) AS [Min-
           МинимальныйЗапас]
       FROM Типы LEFT JOIN Товары
              ON Типы.КодТипа = Товары.КодТипа
GROUP BY  Типы.Категория, Типы.Описание;
```

Завдання 2

Навести текст SQL-запиту, що виконує задану дію.

Сформувати список наступного вида: `Фамилия сотрудника, Общее количество обслуженных им заказов, Последняя дата исполненного заказа, Максимальная стоимость доставки среди всех заказов`, які були обслужені співробітником. Під час формування врахувати співробітників, які не мають заказів, що були обслужені, а також таких, хто має однакові прізвища.

Завдання 3.

Скласти потрібні SQL-запити.

Завдання 3.1. У новій БД створити таблиці **«Викладачі»** та **«Посада»** та заповнити їх відповідними даними.

Задача 3.2. Отримати прізвища викладачів, назви їх посад та норми навантаження, в **SQL**-запиті. Результат помістити у таблицю **«S_Викладач».**

Таблиці **«Викладачі»** та **«Посада»** містять такі дані

Викладачі

Id	Прізвище	Id_Пос	Id_Каф
3	Аврамюк І.С.	01	6
5	Бойко В.М.	02	6
6	Мамай Р.Д.	01	7
11	Сотко В.В.	02	7
15	Оборко Ю.Й.	02	2
17	Купер А.Д.	05	12

Посада

Id_Пос	Посада	Норма
01	професор	500
02	доцент	800
03	ст.викладач	850
04	викладач	900

Завдання 3.3. Скласти запит до таблиць **Викладачі** і **Посада**, при цьому зменшити норму навантаження викладачів на 15% і помістити нове навантаження у поле **Норма_New**.

Завдання 3.4. Виконати запит до таблиць **Викладачі** і **Посада**, при цьому скласти список всіх викладачів, що мають мінімальну норму навантаження (використати для цього підзапит).

Завдання 4

Дати теоретичну довідку з наведеного питання (питання видається викадачем).

ЛІТЕРАТУРА

Базова

1. Дунаев В.В. Базы данных. Язык SQL / В.В.Дунаев. – СПб.: БХВ-Петербург, 2006. – 288 с.
2. Грабер М. Введение в SQL / М. Грабер. - М.: Лори, 1996. – 376 с.
3. Иванов А.А. Базы данных. Языки запросов. Базовый курс: учебное пособие для студентов заочной формы обучения / А.А. Иванов, А.Б. Авербух. – СПб.: СПбГТИ(ТУ), 2011. –45 с.
4. Карпова И.П. Базы данных: Учеб. пособие / И.П. Карпова. – Московский государственный институт электроники и математики. – М., 2009. – 118 с.

Допоміжна

5. Астахова И.Ф., Толстобров А.П., Мельников В.М. SQL в примерах и задачах. – М.: Новое знание, 2002 – 456 с.
6. Грабер М SQL. Справочное руководство – М: Лори, 1997. – 291с
7. Дейт К. Введение в системы баз данных / К. Дейт: Пер. с англ. – М.: Издательский дом "Вильямс", 2001. – 1072с
8. Коннолли Т., Бегг К. Базы данных: проектирование, реализация, сопровождение. Теория и практика, 3-е изд. : Пер. с англ. : Уч. пос. – М.: Изд. Дом "Вильямс", 2003. – 1440 с.
9. Крёнке Д. Теория и практика построения баз данных / Д. Крёнке. – СПб.: Питер, 2003. – 800 с: ил. – (Серия «Классика computer science»).
10. Молинаро Э. SQL Сборник рецептов. – М.: Apress, 2009 – 672с.
11. Ролланд Ф.Д. Основные концепции баз данных / Ф.Д. Ролланд // М.-СПб-Киев: «Вильямс», 2002. –

Інформаційні ресурси

12. Кириллов В.В. Структуризованный язык запросов (SQL): Учебное пособие / В.В. Кириллов, Г.Ю. Громов – СПб.: ИТМО, 1994 [Електронний ресурс]. – Режим доступа: http://citforum.ru/database/sql_kg/
13. Кузнецов С.Д. Введение в стандарты языка баз данных SQL / С.Д.Кузнецов // [Електронний ресурс]. – Режим доступа: http://citforum.ru/database/sqlbook/

ТЕРМІНОЛОГІЧНИЙ СЛОВНИК

Атрибут (властивість). 1) Стовпець. 2) Елемент кортежа. Характеристика, що описує сутність.

Арність відношення. Довжина кортежа (кількість атрибутів).

База даних. Систематизоване сховище великих об'ємів інформації однакової структури та означеної предметної галузі, до якого мають доступ багато прикладних програм..

Вибірка. Операція реляційної алгебри, яка здійснює відбір усіх рядків таблиці, які задовольняють умові відбору.

Вираз реляційної алгебри. Выраз, який складається із застосування реляційних операцій до таблиць.

Відношення. 1) Таблиця. 2) Підмножина декартова добутка доменів.

Вкладений оператор select. Оператор `select` мови SQL, який утримує інший оператор `select` в реченні `from` або `where`.

Внутрішня схема. Опис фізичного подання даних в системі.

Декартовий добуток. Реляційна операція множення, в результаті виконання якої створюється таблиця, що утримує рядок для кожної комбінації рядку з лівого операнда з рядком з правого операнду. Число рядків у результуючій таблиці дорівнює добутку числа рядків лівого та правого операндів.

Домен. Набір допустимих значень атрибуту.

Екземпляр. Об'єкт або набір значень у БД.

З'єднання за еквівалентністю. Реляційна операція з'єднання, умовою з'єднання в якій є вираз у вигляді рівності.

З'єднання. Реляційна операція множення, поєднуючи таблиці базуючись на умові з'єднання. У результуючі таблиці утримується рядок для кожної пари рядків вихідних таблиць, для якого умова з'єднання істинна.

Залежність часткового ключа. Залежність, ліва частина якої є підмножиною ключа.

Запит за зразком (QBE). Спосіб створення запитів із застосуванням графічного інтерфейсу користувача. Метод QBE був створений корпорацією IBM та залучений до багато засобів роботи з базами даних, у тому числі до СУБД Microsoft Access.

Запит. Вимога на отримання інформаційного змісту з бази даних.

Зв'язок. Асоціація між двома або більше сутностями.

Значення `null`. Спеціальне значення атрибуту, відмінне від будь-якого значення з галузі припустимих значень. Значення атрибуту `null` має неоднозначний смисл. Воно може означати значення, яке відсутнє, невідоме значення або незастосовувальний до сутності атрибут.

Значення атрибуту. Значння певного атрибуту однієї сутності.

Зовнішній ключ (foreign key). Атрибут підпорядкованого (дочірнього) відношення, яке є копією первинного (primary key)або унікального (unique) ключа батьківського відношення.

Ключ. Мінімальний набір атрибутів таблиці, який визначає унікальність кожної сутності (рядку).

Конструкці `where`. Конструкція оператора `select` мови SQL, яка задає умову, що визначає, чи будуть певні рядки включені до підсумкової таблиці. Конструкція `where` являє операцію вибірки запиту, а також часто включає у себе умову з'єднання.

Конструкція `from`. Конструкція оператора `select` мови SQL, в якому задаються вихідні таблиці оператора (тобто таблиці, з яких отримують данні). Якщо у конструкції `from` вказано декілька вихідних таблиць, то подана операція є операцією множення цих таблиць. Умови з'єднання визначаються у конструкції `where` або в реченні `on`, якщо використовується ключове слово `join`.

Конструкція `group by`. Конструкція оператора `select` мови SQL, яка визначає, як будуть групуватися відібрані рядки. Для кожної групи відібраних рядків створюється єдиний результуючий рядок.

Конструкція `having`. Конструкція оператора `select` мови SQL, яка задає умову відбору для груп. Конструкція `having` завжди зустрічається разом із конструкцією `group by`.

Конструкція `references`. Конструкція оператора створення таблиці мови SQL, яка декларує, що множина атрибутів є зовнішнім ключем, який посилається на певну таблицю.

Конструкція `select`. Конструкція оператора `select` мови SQL, яка задає в операторі операцію проекції. В конструкції `select` перераховуються атрибути, які повинні бути включені до результуючої таблиці.

Концептуальна модель даних. Опис системи у вигляді, який зрозумілий користувачу.

Кортеж. 1) Рядок. 2) Елемент відношення. Впорядкований список значень атрибутів реляційної схеми.

Логічна модель даних. Опис структури системи баз даних.

Логічна схема. Визначення інформаційного змісту системи зручним для створення бази даних способом.

Множення. Операція реляційної алгебри, об'єднуюча дві таблиці, в результаті якої створюється нова таблиця, атрибутами котрої є атрибути вихідних таблиць. З'єднання та декартовий добуток є операціями множення.

Мова SQL (Structured Query language). Стандартна мова взаємодії з РСУБД.

Мова SQL2/SQL-92. Друга версія мови SQL, стандартизована у 1992 році.

Мова визначення даних (DDL) (Data Definition Language). Мова для визначення концептуальної схеми БД

Мова визначення даних (DDL). Мова визначення структури компонент системи з базою даних.

Мова визначення збереження даних (SDL). Мова визначення внутрішньої схеми БД.

Мова визначення об'єктів (ODL). Стандартна мова для визначення об'єктно-орієнтованих концептуальних схем. ODL була розроблена групою Object Database Management Group (ODMG).

Мова визначення подання (VDL). Мова для визначення зовнішнього подання БД.

Мова керування даними (DML). Мова для визначення операцій, які здійснюють вилучення, включення та поновлення змісту системи з базою даних.

Мова структурованих запитів (SQL). Стандартизована ANSI мова для визначення та керування реляційними базами даних. *SQL* включає у себе як мову визначення даних, так і мову керування даними.

Мова управління даними (DML) (Data Manipulation Language**).** Мова для маніпулювання даними.

Модель «сутність–зв'язок» (ER). Метод побудови концептуальних моделей даних із застосуванням діаграм, який приділяє основну увагу класам сутностей, типам зв'язків та атрибутам.

Обмеження ключа. Обмеження, що накладається на сутності класу та вимагають, щоб ніякі дві сутності не мали однакові значення певного набору атрибутів. Такий набір атрибутів виконує функцію ключа класу.

Обмеження цілісності. Правила, яким повинн відповідати значення атрибута.

Однозначний атрибут. Атрибут з єдиним, неподільним значенням.

Операції над множинами. Операції реляційної алгебри, які об'єднують таблиці з однаковим шаблоном. Такими операціями є об'єднання, перетин та різниця.

Оператор видалення. SQL-оператор, який визначає видалення рядків з таблиці.

Оператор вставки. SQL-оператор, який задає множину рядків, котрі потрібно додати до таблиці.

Оператор поновлення. SQL-оператор, який задає модифікацію певних атрибутів певних рядків таблиці.

Оператор створення таблиці. SQL-оператор, який визначає створення нової таблиці з певними атрибутами та обмеженнями.

Оператор узагальнення. Функція, що синтезує єдине значення із множини рядків таблиці. У мові SQL існують оператори узагальнень: **`avg, count, max, min, sum`.**

Оптимізація запитів. Стратегія підвищення продуктивності виконання запитів, яка застосовує алгебраїчні характеристики реляційних операторів.

Посилальна цілісність. Обмеження, що накладається на таблицю, та вимагає, щоб значення зовнішнього ключа відображало реально існуючу у пов'язаній таблиці сутність.

Первинний ключ. Атрибут (група атрибутів), значення якого є унікальними в межах відношення, ідентифікують кортеж.

Потужність відношення. Кількість кортежів.

Природнє з'єднання. Реляційна операція з'єднання, яка здійснює поєднання двох таблиць, що мають загальний атрибут. У результаті природного з'єднання отримується таблиця, що має рядок для кожної пари рядків вихідних таблиць з однаковим значенням загального атрибуту. Загальні атрибути входять до результуючої таблиці один раз.

Проекція. Операція реляційної алгебри, результуюча таблиця якої складається з відібраних стовпців вихідної таблиці. Число рядків результуючої таблиці може бути менша числа рядків вихідної у результаті видалення рядків-дубликатів.

Реляційна алгебра. Набір операцій з відношеннями та множина правил еквівалентності, які сумісно утворюють мову реляційних виразів.

Реляційна алгебра. Сукупність операцій над відношеннями, яка включає до себе вибірку, проекцію, об'єднання та з'єднання. Для даних операцій існує множина правил еквівалентності виразів. Ці правила, в свою чергу, дозволяють оптимізувати алгебраїчні вирази, які представляють запити баз даних.

Реляційна модель. Модель даних, яка визначає усі данні в вигляді відношень, які побудовані на основі декартових добутків деяких доменів атрибутів.

Реляційна система управління базами даних (РСУБД). СУБД, яка реалізує реляційну модель даних. РСУБД традиційно використовують мову SQL в якості мови взаємодії.

Реляційна схема. Визначення домену значень.

Реляційне числення. Непроцедурний метод визначення запитів шляхом опису структури їх результатів.

Система управления базами даних (СУБД). Сукупність програмного забезпечення та сховища даних, яка забезпечує усі необхідні засоби створення та обслуговування баз даних.

Складний атрибут. Атрибут, значення якого складається з набору окремих полів.

Сутність. Об'єкт реального світу, який представляє інтерес для додатка.

Схема бази даних. Сукупність реляційних схем, які визначають базу даних.

Схема відношення. Перелік атрибутів відношення з їх типами та розмірами.

Уявлення (view, огляд). Запит, що зберігається та створюється на основі команди SELECT. Уявлення реально не утримає даних. Запит, який визначає уявлення, виконується тоді, коли до уявлення здійснюється звернення з іншого запиту, наприклад, SELECT, UPDATE та ін.

Фізична модель даних. Опис способу подання логічної моделі даних при збереженні.

Фізична схема. Визначення інформаційного змісту системи в термінах фізичної реалізації.

Функціональна залежність. Ситуація, коли значення однієї множини атрибутів визначають значення іншої множини.

СПИСОК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

БД – база даних

СУБД – система управління базами даних

DCL – Data Control Language (мова правління даними)

DDL – Data Definition Language (мова визначення даних)

DML – Data Manipulation Language (мова маніпулювання даним)

FK – Foreing key (зовнішній ключ)

NULL – «невизначене» значення домену

PK – Primary key (первинний ключ)

SQL – Structured Query Language (мова маніпулювання даними)

$r \cap s$ – перетин відношень r та s

$r \cup s$ – об’єднання відношень r та s

$r \times s$ – декартовий добуток відношень r та s

$\sigma_{A=a}(r)$ – операція вибірки в r кортежів, в яких значення A равно a

$(\pi_X(r))$ – проекція відношення r на атрибут X

$r \div s$ – ділення відношення r на s

$r \rhd\lhd s$ – природнє з’єжнання відношень r та s

$\left(r \underset{i\,\Theta\,j}{\rhd\lhd} s\right)$ – тета-з’єднання відношень r та s за стовпцями i та j

ДОДАТКИ

ДОДАТОК А
ДОВІДКОВІ ВІДОМОСТІ З МОВИ SQL

Нижче наведені деякі відомості, які можуть бути корисними при фрмуванні запитів у MS Access.

Таблиця А.1. Типи даних MS Access

Назва	Опис
Char	Текстове поле. Може мати розмір не більш ніж 255 символів.
Text, Memo	Текст великого розміру (може вміщувати більш 1 млрд. символів).
Logical	Логічний тип. Може приймати одне з двох значень: True или False.
Byte	Ціле в діапазоні від 0 до 255.
Short	Ціле в діапазоні від -32768 до +32767.
Integer, Int, Long	Довге ціле (в діапазоні від -2147483648 до 2147483647).
Single	Число з плаваючою крапкою одинарної точності. Може приймати значення в діапазоні від $-3.4x10^{38}$ до $3.4x10^{38}$.
Double, Number	Число з плаваючою крапкою потвійної точності. Може приймати значення в діапазоні від $-1.8x10^{308}$ до $1.8x10^{308}$.
Date, Time, DateTime	Дата та час.
Currency	Використовується для позначення грошових сум. Запам’ятовуються 11 знаків від десяткової коми та 4 знака праворуч від десяткової коми.
Counter	Довгі цілі з автоматичним приростом.
OLEObject	OLE-об’єкти, створені в інших программах із використанням протокола OLE. Розмір – до 2 Гбайт.

Продовження таблиці А.1.

Назва	Опис
Binary	Будь-який двійковий об’єкт за розміром до 2 Гбайт. Даний тип може бути використаний для зберігання, наприклад, двійкових файлів.

Таблиця А.2. Деякі групові операції

Им’я операції	Опис
Sum	Обчислює суму полів в групі.
Avg	Обчислює середнє значення для полів групи.
Min	Знаходить найменше значення в групі.
Max	Знаходить найбільше значення в групі.
Count	Підраховує кількість елементів в групі. В якості аргумента можна використовувати «*».
First	Повертає перше значення з групи
Last	Повертає останнє значення з групи

Таблиця А.3. Функції оброботки тексту

Функція	Опис
Left(рядок, n)	Повертає n перших символів рядка ліворуч.
Right(рядок, n)	Повертає n перших символів рядка праворуч.
Mid(рядок, n1, n2)	Повертає n2 символів рядку, починаючи з позиції n1.
InStr(рядок1, рядок2)	Номер позиції, з якої рядок2 входить у рядок1.
Ltrim(рядок)	Видаляє пробіли з початку рядка.
Rtrim(рядок)	Видаляє пробіли з кінця рядка.
Trim(рядок)	Видаляє пробіли з кінця та початку рядка.

Таблиця А.4. Функції обробки дати та часу

Функція	Опис
Date()	Повертає поточну дату.
Now()	Повертає поточну дату та час.
DateDiff(інтервал, дата1, дата2)	Визначає різницю між датами. Аргумент «інтервал» визначає спосіб подання різниці:
	“yyyy” – рік, “q” – квартал, “m” – місяць, “y” – день року, “d” – день, “w” – тиждень, “h” – час, “n” – хвилина, “s” – секунда.
DateAdd(інтервал, число, дата)	Майбутня дата, яка знаходиться від вказаної на задане число інтервалів.
Year(дата)	Повертає число – значення року для вказної дати.
Month(дата)	Повертає число – значення місяця для вказної дати.
Day(дата)	Повертає число – значення дня для вказної дати.

ТаблицяА.5. Функції перетворення

Функція	Опис
Str(аргумент)	Перетворює значення оргумента в текстовий рядок.
Val(рядок)	Перетворює рядок в число
Int(число)	Повертає цілу частину числа

ТаблицяА.6. Операції

Операція	Опис
+	Додавання, конкатенація рядків
-	Різниця
*	Множення

Продовження таблиці А.6

Операція	Опис
/	Ділення
=	Дорівнює
<>	Не дорівнює
>	Більше
<	Менше
>=	Більше або дорівнює
<=	Менше або дорівнює
AND	Логічне «І»
OR	Логічне «АБО»
NOT	Логічне заперечення
операція ANY підзапит	Перевірка на відповідність умові будь-якого елемента з підзапиту
операція ALL підзапит	Перевірка на відповідність умові всіх елементів з підзапиту
EXISTS підзапит	Перевірка на існування в підзапиті хоча б одного елементу
аргумент IS NULL	Чи є аргумент порожнім значенням
аргумент IS NOT NULL	Чи є аргумент не пустим значенням
аргумент1 BETWEEN аргумент2 AND аргумент3	Чи знаходиться значення аргумента «аргумент1» між значеннями «аргумент2» та «аргумент3»
аргумент LIKE образец	Перевірка збігу аргумента зі зразком. У зразку може бути присутнім символ «%», який визначає будя-яке число будь-яких символів. Наприклад, вираз ’Access’ LIKE ’A%s’ повинно повертати значення «True» (істина)

ДОДАТОК Б.
СТРУКТУРА ТА ЗМІСТ НАВЧАЛЬНОЇ БАЗИ «МАГАЗИН»

База даних магазину (файл **Shop.mdb**) має наступну структуру: рис. Б.1.

Стисла характеристика таблиць:

- **Product** – містить інформацію про продукт;
- **Deivery** – поставка товарів;
- **Supplier** – інформація про постачальника;
- **Sale** – продаж товарів;
- **Address** – повна адреса;
- **Country** – список країн;
- **Producer** – інформація про виробників товарів;
- **Markup** – інформація про знижки на продукцію: їх назва та відсоток. Наприклад, сезонна знижка, акція тощо;
- **Measurement** – одиниці виміру: скорочена назва та опис;
- **Category** – категорія товару.

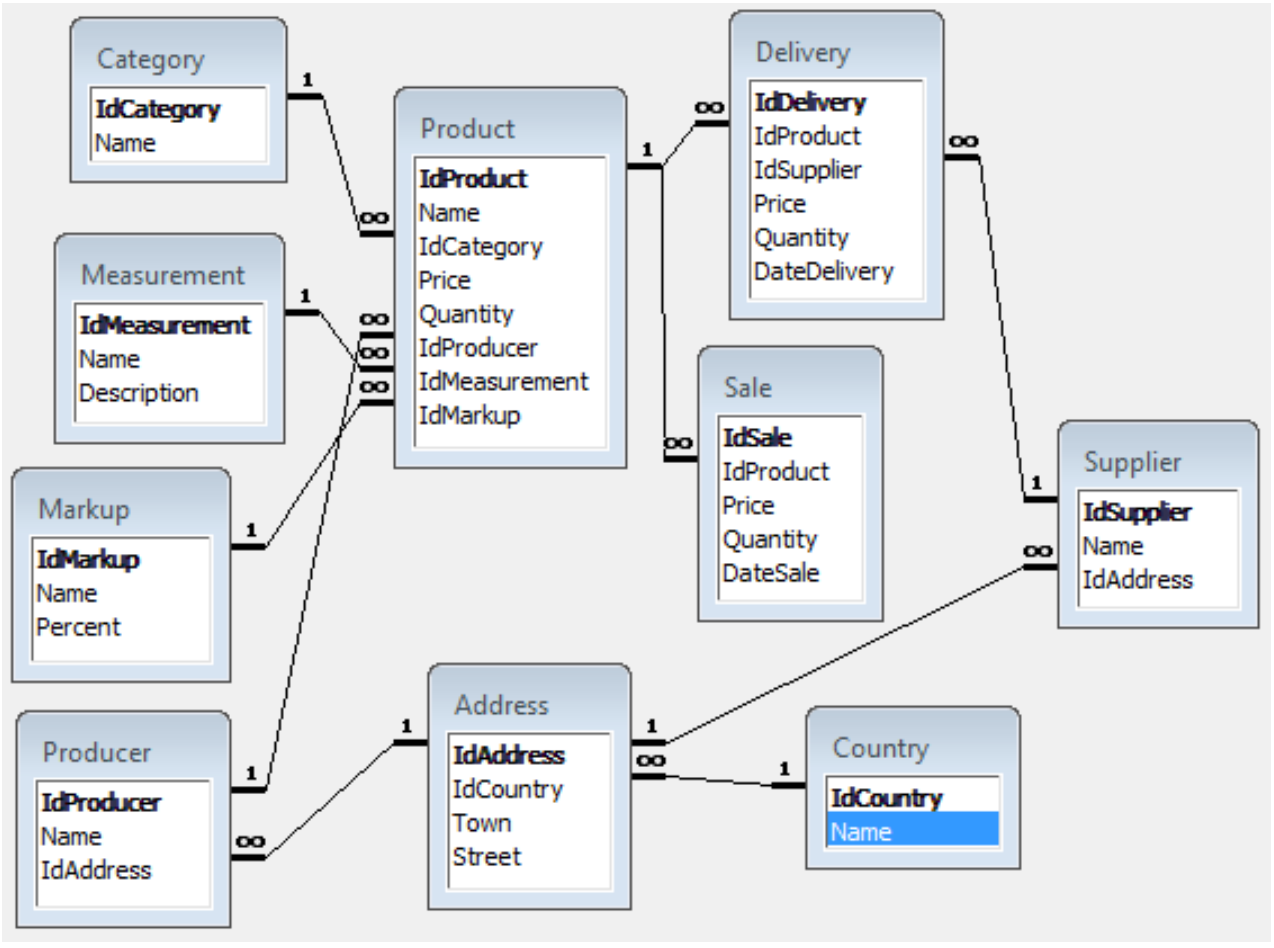


Рис. А.1. Схема БД Shop

Таблиця Б.1. Зміст таблиці Address

Address : таблица

IdAddress	IdCountry	Town	Street
1	US	Вашингтон	NY Street, 154
2	UA	Київ	Артема 41, оф. 32
3	UA	Київ	Будівельників 45, оф. 158
4	UA	Київ	Берегова 108, оф.52
5	UA	Луганськ	Морехідна 3, оф.10
6	RU	Москва	Потьомкінська б.3, оф. 102
7	BR	Мінск	Поштовка 200
8	RU	Москва	Тверська 101
9	UA	Миколаїв	Нікольська 58, оф. 54
10	UA	Миколаїв	Радянська 3, оф. 5
11	UA	Миколаїв	Радянська 15
12	UA	Баштанка	Лугова 23
13	UA	Миколаїв	Радянська
14	UA	Рівне	Паркова 75

Таблиця Б.2. Зміст таблиці Category

Category : таблица

IdCategory	Name
1	Хлібо-булочні вироби
2	СокіВоди
3	Кисломолочні вироби
4	М'ясні вироби
5	Кондитерські вироби
6	Бакалія
7	Фрукти
9	Одяг
10	Оргтехніка
11	Будівельні матеріали
12	Спортивні товари

Таблиця Б.3. Зміст таблиці Country

Country : таблица

IdCountry	Name
BR	Білорусь
CN	Китай
CU	Куба
GB	Англия
PL	Польша
RU	Россия
TR	Турция
UA	Украина
US	США

Таблиця Б.4. Зміст таблиці Delivery

Delivery : таблица

IdDelivery	IdProduct	IdSupplier	Price	Quantity	DateDelivery
3	5	14	2	25	01.09.2014
4	7	10	50	50	10.09.2014
5	10	12	15	10	13.09.2014
6	1	14	2,25	25	13.09.2014
7	6	9	5	22	10.12.2013
8	6	14	5,5	10	08.03.2014

Таблиця Б.5. Зміст таблиці Varkup

Markup : таблица

IdMarkup	Name	Percent
А	Акція	3
П	Поточний продаж	0
Р	Розпродаж	50
С	Сезонна знижка	20

Таблиця Б.6. Зміст таблиці Measurement

Measurement : таблица

IdMeasurement	Name	Description
К01	кг	килограмм
Л01	л	литр
Ш01	шт	количество единиц товара

Таблиця Б.7. Зміст таблиці Producer

Producer : таблица

IdProducer	Name	IdAddress
1	ValeryStyle	2
2	СпортМастер	6
3	Баштанський сир-завод	12
4	ВАТ"Миколаїв-хліб"	11
5	ПП"Ряба курка"	8
6	ЗАТ"Яблуко"	7
7	ВАТ"Картопля"	14
8	ВАТ"Карась"	4
9	ВАТ"ДПС"	5
10	ВАТ"Рум'янец"	4
11	ВАТ"Росинка"	12
12	Banana Republica	1
13	ЗАТ"Ласуня"	13
22	Біола	4
111	Хлібзавод№4	9

Таблиця Б.8. Зміст таблиці Product

Product : таблиця

	IdProduct	Name	IdCategory	Price	Quantity	IdProducer	IdMeasurement	IdMarkup
+	1	Хліб білий	1	4,5	100	111	Ш01	П
+	2	Вода	2	2	2	22	Ш01	П
+	5	Хліб чорний	1	3,5	25	111	Ш01	П
+	6	Ватрушка	1	4,3	20	4	Ш01	П
+	7	Ковбаса"Рум'янець"	4	50	20	5	К01	П
+	8	Фарш"Добрий"	4	64,5	50	5	К01	П
+	9	Цукерки "Радість"	5	43	32	6	К01	П
+	10	Цукерки"Крабіки"	5	16	17	10	К01	П
+	11	Вода"Лімон"	2	12,6	26	8	Л01	П
+	12	Моршинська	6	4,6	5	11	Ш01	П
+	13	Міргородська	6	5,2	5	11	Ш01	П
+	14	Фарш"Добриня"	4	42	50	5	К01	П
+	15	Молоко"Корівка"	3	8,1	75	10	Л01	П
+	16	Йогурт"Живинка"	3	5,4	10	10	Ш01	П
+	17	Сухарі"Житні"	1	6	15	111	К01	П
+	18	Банани	7	15,7	30	12	Ш01	П
+	19	Ковбаса"Лікарська"	4	63,8	15	5	К01	П
+	20	Банани	7	10	10	6	Ш01	А
+	21	Апельсини	7	25,7	110	6	К01	Р

Таблиця Б.9. Зміст таблиці Sale

Sale : таблиця

IdSale	IdProduct	Price	Quantity	DateSale
1	1	3	4	08.09.2014
10	15	5	8	10.09.2014
11	15	4,5	5	11.09.2014
12	18	5	2	16.07.2014
13	15	4,5	1	16.07.2014
14	1	3	2	16.07.2014
15	2	1	1	17.07.2014
16	6	3	1	15.09.2014
17	9	50	2	01.07.2014
18	9	50	3	02.07.2014
19	10	43	2	02.07.2014
2	1	3	7	07.09.2014
20	6	3	2	14.01.2014
3	2	1	5	07.09.2014
4	6	3	5	01.08.2014
5	2	2	2	12.09.2014
6	7	3	5	13.09.2014
7	1	3	1	09.09.2014
8	18	5	9,5	10.07.2014
9	18	4	8	11.07.2014

Таблиця Б.10. Зміст таблиці Supplier

Supplier : таблиця

	IdSupplier	Name	IdAddress
+	1	Banana Republica	1
+	2	ValeryStyle	2
+	3	Баштанський сир-завод	12
+	4	Біола	4
+	5	ВАТ"ДПС"	5
+	6	ВАТ"Картопля"	14
+	7	ВАТ"Карась"	4
+	8	ЗАТ"Ласуня"_WIP	13
+	9	ВАТ"Миколаїв-хліб"	11
+	10	ПП"Ряба курка"	8
+	11	ВАТ"Росинка"	12
+	12	ВАТ"Рум'янец"	4
+	13	СпортМайстер	6
+	14	Хлібзавод№4	9
+	15	ВАТ"Яблуко"	7

ДОДАТОК В

СТРУКТУРА НАВЧАЛЬНОЇ БАЗИ «БОРЕЙ»

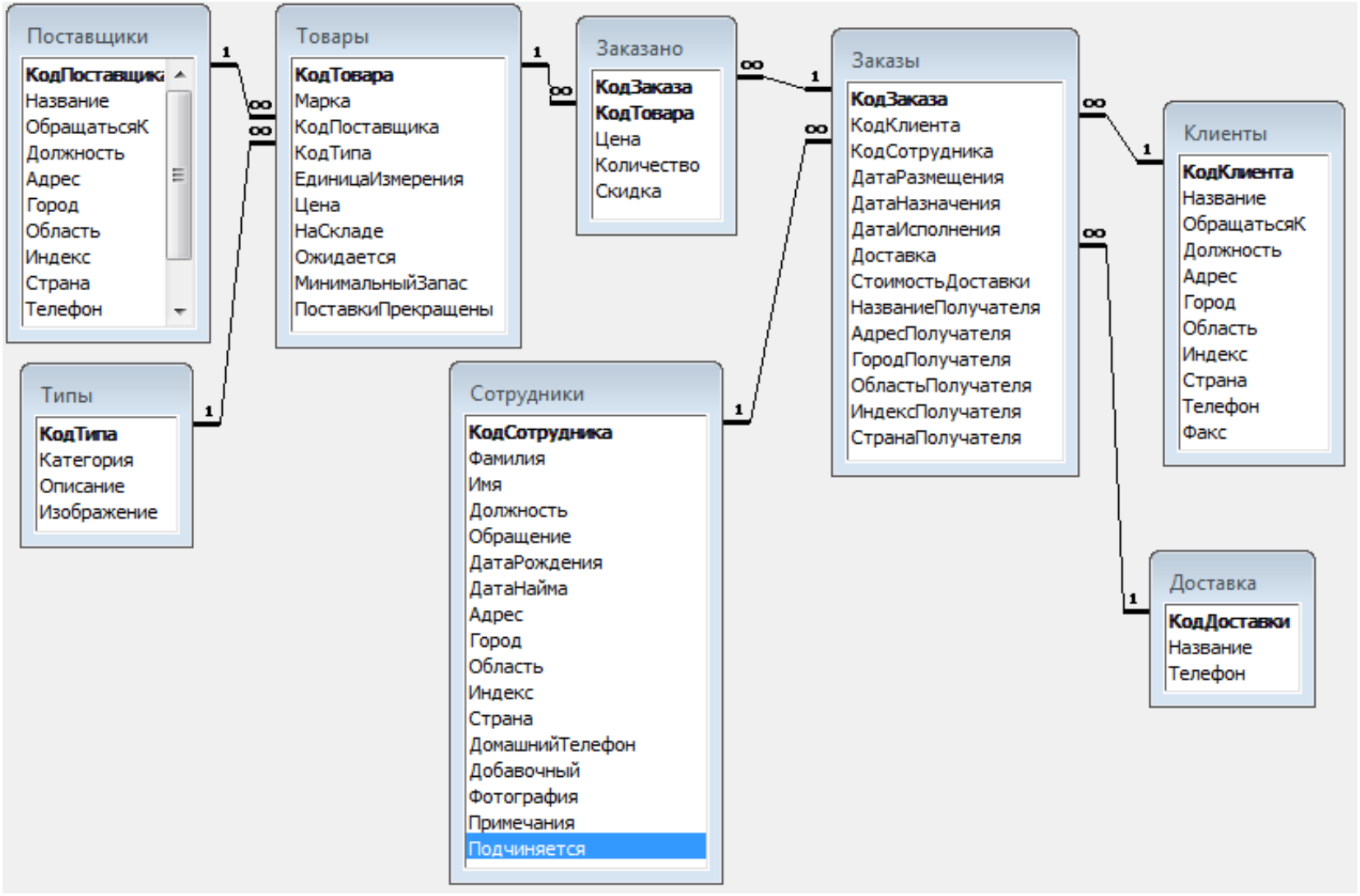


Рис. В.1. Схема БД «Борей»

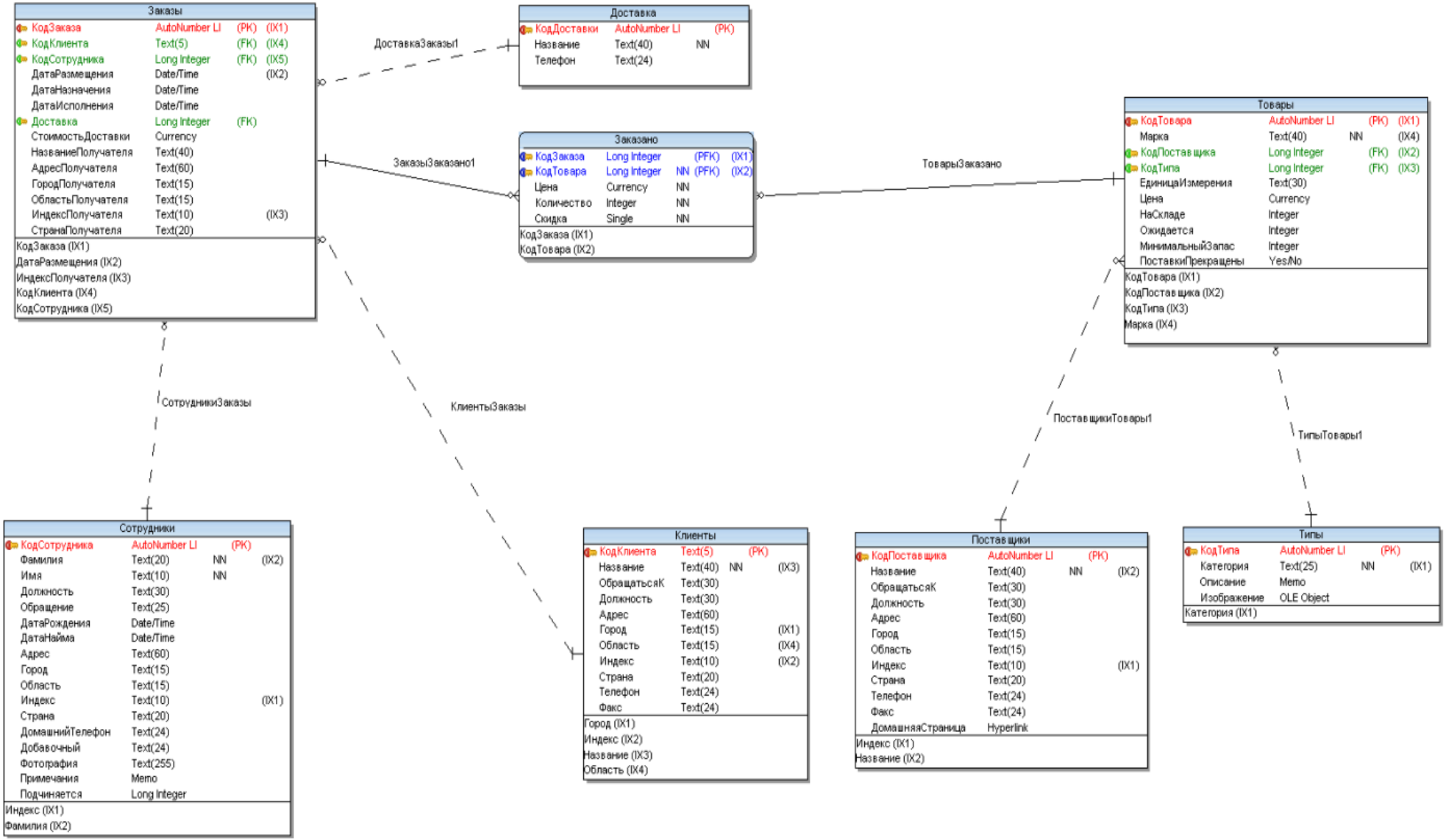


Рис. В.2. Логічна модель БД «Борей»

ДЛЯ НОТАТОК

Навчальний посібник

Ганна Сергіївна Погромська

ПОБУДОВА ЗАПИТІВ НА МОВІ SQL