МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ ІМЕНІ СЕМЕНА КУЗНЕЦЯ

БАЗИ ДАНИХ

Лабораторний практикум

для студентів галузі знань
12 "Інформаційні технології"
першого (бакалаврського) рівня

Харків
ХНЕУ ім. С. Кузнеця
2019

УДК 004.65(07.034)
Б17

Укладачі: В. В. Федько
В. П. Бурдаєв

Затверджено на засіданні кафедри інформаційних систем.
Протокол № 4 від 20.11.2018 р.

Самостійне електронне текстове мережеве видання

Бази даних [Електронний ресурс] : лабораторний практикум
Б17 для студентів галузі знань 12 "Інформаційні технології" першого (бакалаврського) рівня / уклад. В. В. Федько, В. П. Бурдаєв. – Харків : ХНЕУ ім. С. Кузнеця, 2019. – 215 с.

Наведено опис восьми лабораторних робіт із навчальної дисципліни і вимоги до їх виконання. Розглянуто засоби проектування баз даних, створення, експлуатації та аналізу даних на прикладах реляційної бази даних *MS SQL Server* та документоорієнтованої *MongoDB*. Подано критерії оцінювання знань студентів і професійні компетентності, якими має володіти студент після вивчення дисципліни.

Рекомендовано для студентів галузі знань 12 "Інформаційні технології" першого (бакалаврського) рівня.

УДК 004.65(07.034)

Вступ

Лабораторний практикум складений відповідно до програми навчальної дисципліни "Бази даних" для студентів галузі знань 12 "Інформаційні технології" першого (бакалаврського) рівня та спрямований на забезпечення лабораторних занять з даної дисципліни.

Метою практикуму є оволодіння засобами проектування баз даних, створення, експлуатації та аналізу даних. Для цього використовуються як традиційні (реляційні), так і новітні (NoSQL) системи управління базами даних.

Вивчення навчальної дисципліни спрямоване на набуття студентами компетентностей у галузі проектування систем управління базами даних та їх використання, що дозволить майбутнім фахівцям розв'язувати складні економічні задачі у подальшій професійній діяльності.

Необхідним елементом успішного засвоєння навчального матеріалу дисципліни є практична робота студентів з літературою з питань проектування та використання баз даних.

У процесі вивчення навчальної дисципліни основна увага приділяється оволодінню такими професійними **компетентностями**:

уміння розробляти, аналізувати та застосовувати ефективні алгоритми для розв'язання професійних завдань для СУБД;

уміння проектувати логічні та фізичні моделі баз даних;

уміння моделювати системи та процеси, стани та поведінки складних об'єктів інформатизації в процесі розроблення інформаційних систем і технологій;

здатність аналізувати предметну область та її модель;

здатність застосовувати сучасні теорії організації баз даних, методи та технології їх розроблення;

здатність забезпечувати захист даних в інформаційних системах, побудованих на основі концепції баз даних.

У результаті виконання лабораторного практикуму студент отримає такі **вміння** та **навички**:

формулювати вимоги до бази даних і забезпечувати її властивості;

проектувати концептуальну модель даних конкретної предметної області;

застосовувати принципи побудови баз даних ненадлишковості, несуперечності та незалежності даних;

вибирати СУБД у процесі технічного проектування на основі оцінних варіантів баз даних, вимог користувачів, аналізу технічних, економічних, функціональних, сервісних характеристик СУБД, використовуючи науково-технічну та довідкову інформацію;

розробляти логічну структуру бази даних у процесі технічного проектування за допомогою методу нормалізації відношень, використовуючи методи реляційної алгебри, рівні абстракції даних, вимоги вибраної СУБД;

розробляти таблиці баз даних і зв'язок між ними в умовах технічного проектування за допомогою відповідного технічного та програмного забезпечення, використовуючи конструктори таблиць;

розробляти фізичну структуру бази даних у процесі робочого проектування за допомогою вибраної СУБД, використовуючи сучасні технічні і програмні засоби розробника баз даних;

виконувати операції реляційної алгебри;

будувати ER-діаграми CASE-засобами;

створювати таблиці баз даних за допомогою програмних і технічних засобів проектування баз даних, використовуючи візуальні інструменти інтегрованих оболонок розробника програмного забезпечення та засоби мови DDL;

розробляти операції введення, модифікації, вилучення, відображення даних у таблиці бази даних у процесі робочого проектування за допомогою технічних і програмних засобів, використовуючи форми введення та модифікації даних і табличні режими та засоби мови DML;

розробляти засоби навігації по набору даних в умовах доступу до потрібного запису набору даних за допомогою програмних засобів СУБД, використовуючи навігаційні методи об'єктів;

розробляти методи сортування, фільтрації, пошуку даних у процесі відбору потрібних даних, що відповідають будь-яким критеріям, за допомогою програмних засобів СУБД;

установлювати обмеження цілісності засобами мови SQL;

застосовувати заходи захисту даних засобами мови SQL;

здійснювати вибір засобів для побудови систем аналітичної обробки даних;

застосовувати аналітичні функції SQL для розв'язання конкретних економічних задач;

виконувати базові операції з базою даних *MongoDB*.

В основу побудови лабораторного практикуму закладений принцип поступовості вивчення предмету. Першими подані лабораторні роботи, спрямовані на вивчення мови SQL. Під час їх виконання студенти ознайомляться з основними елементами баз даних: таблицями, рядками, стовпцями, типами даних, зв'язками між таблицями, поданнями тощо. Це дозволяє перейти до оволодіння технологіями проектування баз даних на основі сформованих уявлень про структурні елементи, з яких складається база даних. Закріплення навичок проектування баз даних здійснюється під час вивчення теми "Аналіз даних" для побудови сховищ даних. Для заповнення сховища значними обсягами даних використовуються засоби SQL. Усі розглянуті вище елементи реляційних баз даних використовуються з метою оволодіння навичками роботи з базою даних *NoSQL* як основи для порівняння.

Лабораторний практикум складається з опису восьми лабораторних робіт.

Перша лабораторна робота присвячена набуттю практичних навичок роботи з базами даних у середовищі *Visual Studio*, створення баз даних і таблиць візуальними засобами та засобами мови DDL. Після виконання лабораторної роботи студент умітиме самостійно створювати бази даних, таблиці в них і заповнювати таблиці даними.

Метою другої лабораторної роботи є набуття практичних навичок побудови запитів на отримання даних таких типів: детальні, групування, підзапити, модифікування даних і створення та використання подань. За інтелектуальним навантаженням ця робота є однією з найбільш насичених. Потрібно подолати психологічний бар'єр між алгоритмічними та декларативними засобами обробки даних. Після виконання лабораторної роботи студент повинен уміти самостійно будувати детальні запити, запити, що містять групування даних за співпаданням значень указаних полів, а також запити, до складу яких входять підзапити, запити на модифікування даних, створювати подання як об'єкти бази даних.

Третя лабораторна робота, на відміну від другої, спрямована на побудову містка між декларативними й алгоритмічними засобами обробки даних. Під час її виконання студент: ознайомиться з алгоритмічними розширеннями мови SQL на прикладі мови Transact SQL; набуде практичних навичок побудови збережених процедур і функцій користувача, створення і використання тригерів; навчиться їх тестувати.

Четверта лабораторна робота присвячена набуттю практичних навичок нормалізації реляційної бази даних, освоєнню операцій реляційної алгебри, теоретико-множинних і спеціальних реляційних операцій. Перша частина роботи є найбільш креативною у лабораторному практикумі. Під час її виконання не використовуються складні програмні засоби. Достатньо тільки текстового редактора Word, щоб оформити результати проектування. Проблеми виникають з обґрунтуванням переходів від більш низької нормальної форми до більш високої, особливо у тому разі, коли база даних уже перебуває у більш високій нормальній формі та попередні кроки виконувати не потрібно. За таких обставин студенту пропонується дещо змінити постановку задачі, щоб усі попередні кроки стали необхідними. Такий підхід забезпечує добре оволодіння технологією нормалізації, яка є центральною ланкою під час проектування бази даних.

П'ята лабораторна робота продовжує тему проектування баз даних, але, на відміну від четвертої, у ній широко використовуються CASE-засоби, а саме – програма *ERwin Data Modeler*. Під час виконання роботи студент навчиться: аналізувати предметну область і синтезувати її модель з використанням CASE-технологій; самостійно формувати прості та складні ER-моделі предметної області; проводити пряме та зворотне проектування схеми бази даних; визначати стратегії підтримки посилальної цілісності бази даних; формувати звіти за сформованою моделлю предметної області.

Шоста лабораторна робота спрямована на: набуття практичних навичок роботи зі сховищами даних у середовищі *Visual Studio* або *Management Studio*; вироблення умінь із створення зведених таблиць і діаграм у середовищі *Excel* на основі даних сховища SQL Server; оволодіння навичками проектування, створення та заповнення сховища даних засобами мови SQL. Після виконання лабораторної роботи студент навчиться самостійно проектувати та створювати сховища даних, заповнювати оперативні бази і сховища даних значною кількістю даних, виконувати аналіз даних у середовищі *Excel* на основі даних сховища.

Цілями сьомої лабораторної роботи є: набуття практичних навичок установлення СУБД *MongoDB*, роботи в оболонці цієї СУБД; вироблення умінь додавання, зміни та видалення даних у базі даних *MongoDB*; отримання практичних навичок побудови запитів на вибірку в документо-орієнтованій базі даних, а також проектування документо-орієнтованої бази даних.

Ціллю восьмої лабораторної роботи є набуття практичних навичок: підключення до порталу Microsoft Azure, створення і розгортання бази даних SQL засобами хмарної технології *Microsoft Azure* і міграції локальної бази даних з середовища *Visual Studio* або *Management Studio* на портал *Microsoft Azure*. Після виконання лабораторної роботи студент буде вміти: самостійно заходити до порталу *Microsoft Azure*; створювати нову базу даних і здійснювати її розгортання в *Microsoft Azure*; створювати таблиці та запити для бази даних на порталі *Microsoft Azure* та виконувати міграцію локальної бази даних на порталі *Microsoft Azure*.

Завдання лабораторних робіт мають бути посильними для розуміння студентом. З цією метою кожний студент самостійно вибирає рівень складності з-поміж таких: фронтальний; індивідуальний; компетентнісний.

Якщо вибрано *фронтальний рівень*, то студент виконує завдання базового рівня, що детально описане в інструкції. За його виконання студент отримує чотири бали за дванадцятибальною системою оцінювання. Такі завдання детально описано в практикумі.

З метою випробування своїх сил і підвищення оцінки студент може самостійно розв'язати ще кілька задач, які подано в розділі "Завдання для самостійного виконання". Ним закінчується опис кожної лабораторної роботи. Частина завдань є репродуктивного, а інші – креативного типу. За правильне розв'язання кожного репродуктивного завдання додається ще один бал, а компетентнісного – до 5 балів. До отриманої суми балів студент може додати ще два бали, якщо самостійно запропонує і розв'яже оригінальну задачу за темою, що вивчається. Ця задача має бути з предметної галузі навчання чи майбутньої професії студента. Загальна оцінка за цим рівнем не перевищує восьми балів.

У разі вибору *індивідуального рівня* студент ознайомлюється з інструкцією щодо виконання завдання базового рівня і розв'язує аналогічну задачу з множини варіантів, поданих в інструкції. За виконання такого індивідуального завдання студент отримує шість балів. Ще два бали він може отримати, якщо адаптує до предметної галузі обраного варіанта задачі, які подані в інструкції, і розв'яже їх. Подібно до фронтального рівня студент може додати ще два бали до отриманої суми балів, якщо сформулює та розв'яже оригінальну задачу за темою, що вивчається. Загальна оцінка за цим рівнем не перевищує десяти балів.

На *компетентнісному рівні* студент демонструє можливість самостійно ставити та розв'язувати задачі за темою, що вивчається, з предметної галузі навчання чи майбутньої професії. Спочатку він формулює і розв'язує задачу, аналогічну базовій (сім балів), потім – аналогічну додатковим задачам (ще два бали) та насамкінець – оригінальну задачу (до трьох балів). Загальна оцінка за цим рівнем може досягати дванадцяти балів.

За запізнення із захистом лабораторної роботи знімається 2 бали за кожний тиждень.

Отримана кількість балів за відповіді на кожне завдання лабораторної роботи підсумовується. У результаті такого підрахунку студентом може бути отримано від 0 до 12 балів.

Для зручності переміщення матеріалом лабораторного практикуму передбачені закладки, гіперпосилання, нумератор сторінок. У програмі Adobe Acrobat Reader це кнопка **Закладки** .

В інструкціях до виконання лабораторних робіт є тексти фрагментів скриптів. Їх можна копіювати в буфер, а потім вставляти в редактор запитів Transact SQL в середовищі *Visual Studio*. Таким прийомом рекомендується користуватися, коли скрипт абсолютно зрозумілий. Якщо ж в ньому потрібно розібратися, краще вводити текст скрипта самостійно. Під час введення, як правило, стає зрозумілим алгоритм, розміщений в скрипті.

За результатами виконання роботи студент має оформити електронний звіт засобами Word. З кожного завдання потрібно записати формулювання задачі, подати текст команд і зробити скриншот з результатами виконання. У висновках з лабораторної роботи слід дати самооцінку тому, які нові знання, вміння та навички отримано під час виконання роботи.

Лабораторна робота 1
Створення баз даних і таблиць у *SQL Server*

Цілі роботи

1. Набуття практичних навичок роботи з базами даних у середовищі *Visual Studio* або *Management Studio*.

2. Набуття практичних навичок зі створення баз даних і таблиць візуальними засобами.

3. Набуття практичних навичок зі створення баз даних і таблиць засобами мови DDL.

Перед виконанням роботи студент повинен знати:

основи використання *Visual Studio;*
основні об'єкти бази даних *MS SQL Server*;
основні команди мови SQL.

Після виконання лабораторної роботи студент повинен уміти:

самостійно створювати бази даних, таблиці в них;
заповнювати таблиці даними.

Підготовча частина

Хід роботи

1.1. Створення бази даних у вигляді mdf-файла у середовищі *Visual Studio.*

1.2. Створення таблиць візуальними засобами *Visual Studio* та встановлення зв'язків між ними.

1.3. Заповнення таблиць даними з використанням візуальних засобів.

1.4. Створення бази даних засобами мови DDL.

1.5. Створення таблиць засобами мови DDL.

1.6. Заповнення таблиць засобами мови SQL.

1.7. Створення індивідуальної бази даних. Створення і заповнення таблиць засобами мови SQL

Форма звітності

За результатами виконання роботи необхідно оформити електронний звіт засобами Word. За кожним завданням слід зробити скриншоти з результатами виконання, а за завданнями 1.4 – 1.6 ще додатково

записати відповідні SQL-скрипти. За завданням 1.7 помістити формулювання вибраного варіанту, тексти завдань, запитів на їх виконання і скриншоти з результатами. Аналогічним чином оформити звіт з виконання кожного завдання із п. "Завдання для самостійного виконання".

Критерії оцінювання

У 12-бальній системі оцінка результату захисту лабораторної роботи формується за такими правилами:

1. За кожне завдання з діапазону 1.1 – 1.6 може бути виставлено від 0 до 1 бала.

2. За завдання 1.7 може бути виставлено від 0 до 3 балів.

3. За самостійне формулювання схеми бази даних під час виконання завдання 1.7 додається 1 бал.

4. За кожне завдання із п. "Завдання для самостійного виконання" може бути виставлено від 0 до 1 бала.

5. За кілька варіантів розв'язання одного із завдань додається 1 бал.

6. За вибір варіанту, який з кількох варіантів розв'язання є оптимальним, та обґрунтування вибору додається 1 бал.

7. За побудову скрипта мовою SQL, в якому описані всі операції з базою даних (створення бази даних, таблиць і їхнє заповнення), додається 1 бал за кожний скрипт.

8. За створення бази даних, таблиць і їхнє заповнення в іншій СУБД (Oracle, DB2 тощо) додається 1 бал за кожну базу даних.

9. За запізнення із захистом лабораторної роботи знімається 2 бали за кожний тиждень.

Отримана кількість балів з відповідей на кожне завдання лабораторної роботи підсумовується. У результаті такого підрахунку студент може отримати від 0 до 12 балів.

Рекомендована література: [1; 2; 6; 8; 12; 13].

Основні поняття

Structured Query Language (SQL) – це непроцедурна мова, яка використовується для управління даними реляційних СУБД.

Термін "непроцедурна" означає, що мовою можна сформулювати те, що потрібно зробити з даними, але не можна проінструктувати, як його слід виконати. У стандарті мови SQL відсутні алгоритмічні конструкції

такі, як мітки, оператори циклу, умовні переходи тощо. Хоча в діалектах мови, що створені різними розробниками (Microsoft, Oracle, IBM) вони є.

Мова SQL має п'ять основних типів команд, кожну з яких можна розглядати як окрему мову. До них входять такі:

DDL (Data Definition Language) – мова визначення даних. Призначена для створення, змінення та вилучення об'єктів;

DML (Data Manipulation Language) – мова маніпуляцій даними. Містить оператори, що дозволяють вибирати, додавати, видаляти та модифікувати дані;

DCL (Data Control Language) – мова управління даними. Застосовується для реалізації адміністративних функцій, які надають або скасовують право (привілей) використовувати базу даних, таблиці в базі даних, а також виконувати ті чи інші оператори SQL;

TCL (Transaction Control Language) – мова управління транзакціями. Дозволяє організовувати роботу транзакцій, що здійснюють зміни у базі даних за допомогою груп операторів DML;

CCL (Cursor Control Language) – мова управління курсором. Містить оператори визначення курсора, підготовки SQL-речень для виконання, а також для деяких інших операторів.

Для створення нової бази даних засобами мови SQL слід використовувати таку команду:

```
CREATE DATABASE Ім'я_бази_даних [Параметри];
```

Перелік і можливі значення параметрів залежать від конкретної СУБД і найчастіше описують фізичні параметри бази даних, засоби управління безпекою, мовні параметри тощо.

Приклад 1. *Створення бази*. Створіть базу даних Sales в Microsoft SQL Server:

```
USE master;
GO
CREATE DATABASE Sales
ON
( NAME = Sales_dat,
    FILENAME = 'C:\Program Files\Microsoft SQL
Server\MSSQL12.MSSQLSERVER\MSSQL\DATA\saledat.mdf',
    SIZE = 10,
    MAXSIZE = 50,
```

```
    FILEGROWTH = 5 )
LOG ON
( NAME = Sales_log,
    FILENAME = 'C:\Program Files\Microsoft SQL
Server\MSSQL12.MSSQLSERVER\MSSQL\DATA\salelog.ldf',
    SIZE = 5MB,
    MAXSIZE = 25MB,
    FILEGROWTH = 5MB ) ;
GO
```

Базовий синтаксис оператора створення таблиці має такий вигляд:

```
CREATE TABLE ім'я_таблиці
    (ім'я_стовпця тип_даних [ PRIMARY KEY] [NULL | NOT NULL ] [,...n])
```

Приклад 2. *Створення таблиці*. Створіть таблицю Товари з автоінкрементним ключем.

```
CREATE TABLE Товари(
    Код_товару int IDENTITY(1,1) PRIMARY KEY,
    Товар nvarchar(25) NOT NULL,
    Код_групи int NULL,
    Ціна money NULL,
    Ціна_закупівлі money NULL
);
```

Більш розширений синтаксис команди на створення таблиць у базі даних має такий вигляд:

```
CREATE TABLE Ім'я_таблиці
    {(ім'я_стовпця тип_даних [ NOT NULL ][ UNIQUE]
        [DEFAULT <значення>]
        [ CHECK (<умова_вибору>)][,...n]}
        [CONSTRAINT ім'я_обмеження]
        [PRIMARY KEY (ім'я_стовпця[,...n])
    {[UNIQUE (ім'я_стовпця[,...n])}
    [FOREIGN KEY (ім'я_стовпця_зовнішнього_ключа[,...n])
        REFERENCES ім'я_батьківської_таблиці
        [(ім'я_стовпця_батьківської_таблиці[,...n])],
        [MATCH {PARTIAL | FULL}]
        [ON UPDATE {CASCADE| SET NULL |SET DEFAULT |NO ACTION}]
        [ON DELETE {CASCADE| SET NULL |SET DEFAULT |NO ACTION}]
    {[CHECK(<умова_вибору>)][,...n]})
```

Приклад 3. *Створення таблиці з обмеженнями*. Створіть таблицю Продажі з обмеженням посилальної цілісності та унікальності та авто-інкрементним ключем.

```
CREATE TABLE Продажі(
    Код_продажу int IDENTITY(1,1) PRIMARY KEY,
    Дата date NULL,
    Код_товару int NOT NULL REFERENCES Товари
        ON UPDATE CASCADE,
    Код_виробника int NULL REFERENCES Виробники
        ON DELETE SET NULL ON UPDATE CASCADE,
    Кількість smallint NOT NULL CHECK (Кількість >0 ),
    CONSTRAINT U1_PRODAGI
        UNIQUE (Код_товару, Код_виробника, Дата)
);
```

Коли необхідно зробити певні корективи у структурі вже створеної таблиці, використовують команду ALTER TABLE. Вона може виконувати такі дії:

додавати у таблицю новий стовпець;
видаляти стовпець з таблиці;
додавати у визначення таблиці нове обмеження;
видаляти з визначення таблиці існуюче обмеження;
задавати для стовпця значення за замовчуванням;
відміняти для стовпця значення за замовчуванням.

Команда ALTER TABLE має такий формат:

```
ALTER TABLE Ім'я_таблиці
    [ADD [COLUMN] ім'я_стовпця тип_даних
    [ NOT NULL ] [UNIQUE]
    [DEFAULT <значення>] [ CHECK (<умова_вибору>)]]
    [DROP [COLUMN] ім'я_стовпця [RESTRICT | CASCADE ]]
    [ADD [CONSTRAINT [ім'я_обмеження]]
    [{PRIMARY KEY (ім'я_стовпця [,...n])
        |[UNIQUE (ім'я_стовпця [,...n])}
        |[FOREIGN KEY (ім'я_ стовпця_зовнішнього_ключа [,...n])
    REFERENCES ім'я_батьківської_таблиці
        [(ім'я_ стовпця _ батьківської_таблиці [,...n])],
    [ MATCH {PARTIAL | FULL}
    [ON UPDATE {CASCADE| SET NULL |
        SET DEFAULT | NO ACTION}]
    [ON DELETE {CASCADE| SET NULL |
        SET DEFAULT | NO ACTION}]
        |[CHECK(<умова_вибору>)][,...n]}]
    [DROP CONSTRAINT ім'я_обмеження
```

```
        [RESTRICT | CASCADE]]
    [ALTER [COLUMN] SET DEFAULT <значення>]
    [ALTER [COLUMN] DROP DEFAULT]
```

Приклад 4. *Додавання стовпця*. Додайте у таблицю Товар новий стовпець Сорт.

```
ALTER TABLE Товари
    ADD Сорт integer;
```

Приклад 5. *Додавання обмеження*. Додайте обмеження на допустимі значення стовпця Сорт.

```
ALTER TABLE Товари
    ADD CONSTRAINT CH_Sort CHECK (Сорт >=0 AND Сорт =<3);
```

Приклад 6. *Видалення обмеження*. Видаліть обмеження унікальності значень комбінації полів Код_товару, Код_виробника та Дата з таблиці Продажі.

```
ALTER TABLE Продажі
    DROP CONSTRAINT U1_PRODAGI;
```

Для видалення об'єктів використовується команда DROP, після якої йде назва виду об'єкта. Наприклад: TABLE – таблиця, INDEX – індекс, CONSTRAINT – обмеження, а потім – назва самого об'єкта.

Приклад 7. *Видалення таблиці.* Видаліть з бази даних таблицю Товари.

```
DROP TABLE Товари;
```

Подання (VIEW) – це тимчасові (інакше – віртуальні) таблиці. Вони є об'єктами бази даних, інформація в яких не зберігається постійно, як в базових таблицях, а динамічно формується зі зверненням до них. Це фактично той самий запит, який виконується кожний раз за участю в якій-небудь команді. Застосування подань дозволяє розробнику бази даних забезпечити кожному користувачеві або групі користувачів найбільш доцільні способи роботи з даними. Це вирішує проблему простоти їх використання та безпеки.

Подання створюють за таким форматом:

```
CREATE VIEW Ім'я_подання [(ім'я_стовпця [,...n])]
    [WITH ENCRYPTION]
    AS SELECT_команда [WITH CHECK OPTION]
```

Приклад 8. *Створення подання*. Створіть подання Прайс на основі таблиці Товари.

```
CREATE VIEW Прайс
    AS SELECT Товар, Ціна, Ціна * 0.2 AS Податок
        FROM Товари
```

Подання Прайс має три стовпці. Його можна використовувати майже в тих самих випадках, що й таблицю Товари. Винятком є та обставина, що за його допомогою не можна змінювати дані в таблиці Товари, оскільки подання має обчислюване поле Податок.

Подання можна змінювати командою ALTER і видаляти командою DROP.

Практична частина

1.1. Створення бази даних у вигляді mdf-файла у середовищі *Visual Studio*

Завдання

Створіть базу даних ***ХлібПрізвище*** в СУБД MS SQL Server у вигляді mdf-файла.

Примітки.

1) замість слова ***Прізвище*** в значенні імені бази даних потрібно вставити своє прізвище. Тоді база даних матиме відповідне ім'я, наприклад, ***ХлібПетренко***;

2) в описах подальших завдань база даних буде мати узагальнене ім'я ***Хліб***.

Виконання

1. Відкрийте Visual Studio.
2. Створіть проект Windows Forms мовою C# з ім'ям ***appПрізвище***.

Примітка.

Замість слова ***Прізвище*** в значенні імені проекту вставте своє прізвище. Тоді проект матиме відповідне ім'я, наприклад ***appПетренко***.

3. Додайте до проекту mdf-файл бази даних.

Для цього:

3.1. Клацніть правою клавішею миші (ПКМ) на значку проекту у вікні **Solution Explorer** і з контекстового меню виберіть команду **Add – New Item**.

3.2. Виберіть елемент **Data** у лівій панелі вікна **Add New Item**, потім елемент **Service-base Database** і в полі **Name** введіть ім'я бази даних, наприклад ***ХлібПетренко.mdf***. Після цього клацніть кнопку **Add** (рис. 1.1).

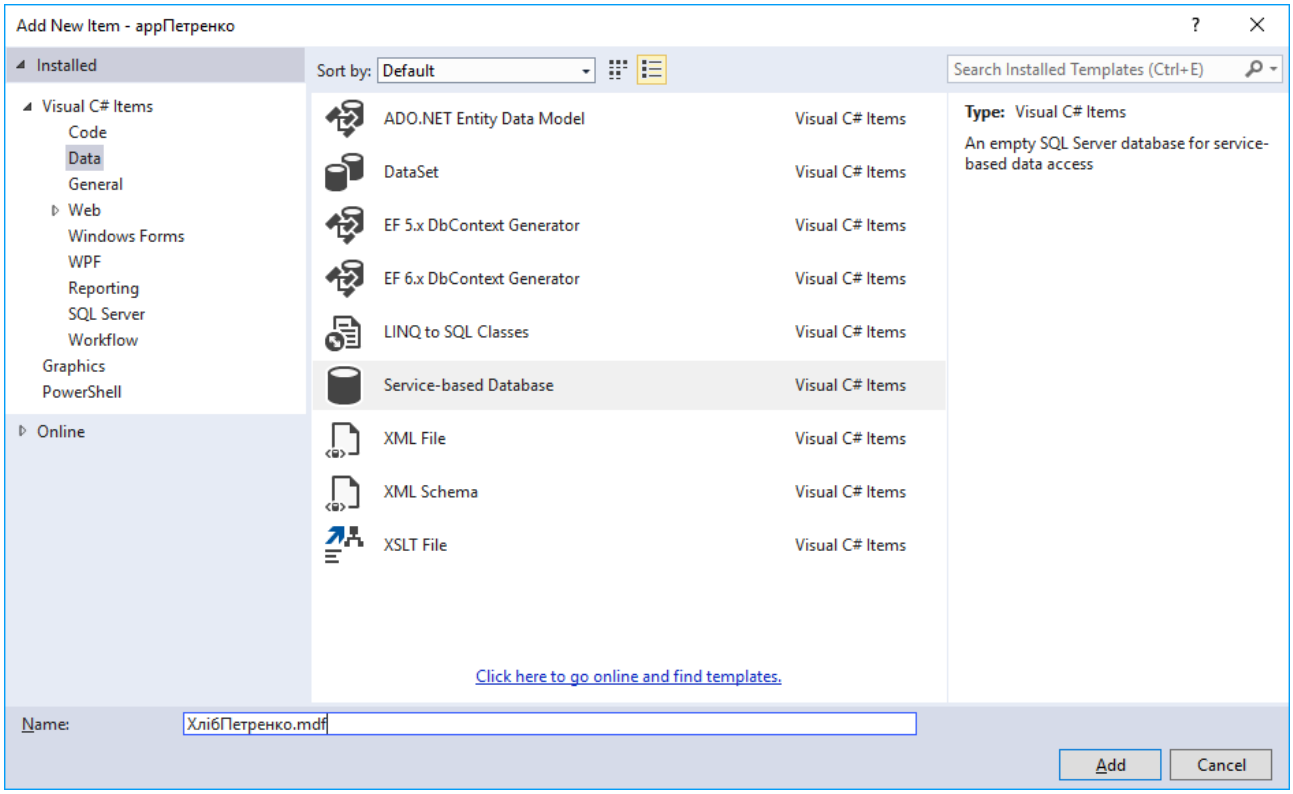


Рис. 1.1. **Вікно Add New Item**

4. Зачекайте деякий час, поки у вікнах **Solution Explorer** і **Server Explorer** з'являться значки бази даних (рис. 1.2, 1.3).

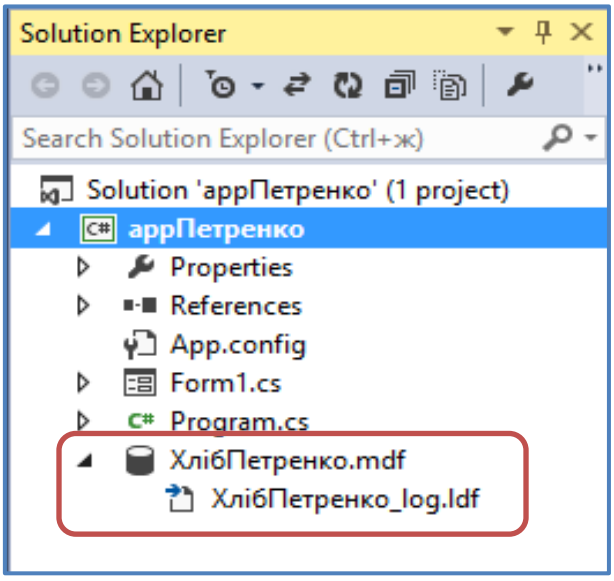


Рис. 1.2. **Значок бази даних у вікні Solution Explorer**

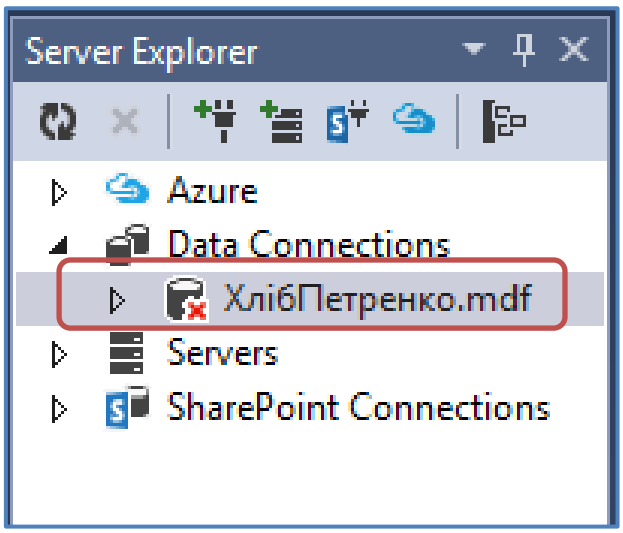


Рис. 1.3. **Значок бази даних у вікні Server Explorer**

Примітка. Якщо не з'являється значок бази даних у вікні **Server Explorer**, клацніть значок бази даних у вікні **Solution Explorer**.

5. У звіті з лабораторної роботи помістіть текст завдання і скриншоти, аналогічні рис. 1.2, 1.3.

6. Відкрийте вузол значка бази даних у вікні **Server Explorer**, а в ньому підвузол **Tables**. Ця папка виявиться порожньою. Таблиці в ній створюються у наступному завданні.

1.2. Створення таблиць візуальними засобами *Visual Studio* та встановлення зв'язків між ними

Постановка задачі

База даних ***Хліб*** містить такі таблиці:

Товари (Код_товару, Товар, Ціна, Ціна_закупівлі);

Виробники (Код_виробника, Виробник, Адреса, Телефон);

Продажі (Код_продажу, Дата, Код_виробника, Код_товару, Кількість).

Потрібно створити таблиці у базі даних ***Хліб*** і з'єднати їх (рис. 1.4).

Завдання 1

Створіть таблицю ***Товари*** у базі даних ***Хліб*** з використанням візуальних засобів.

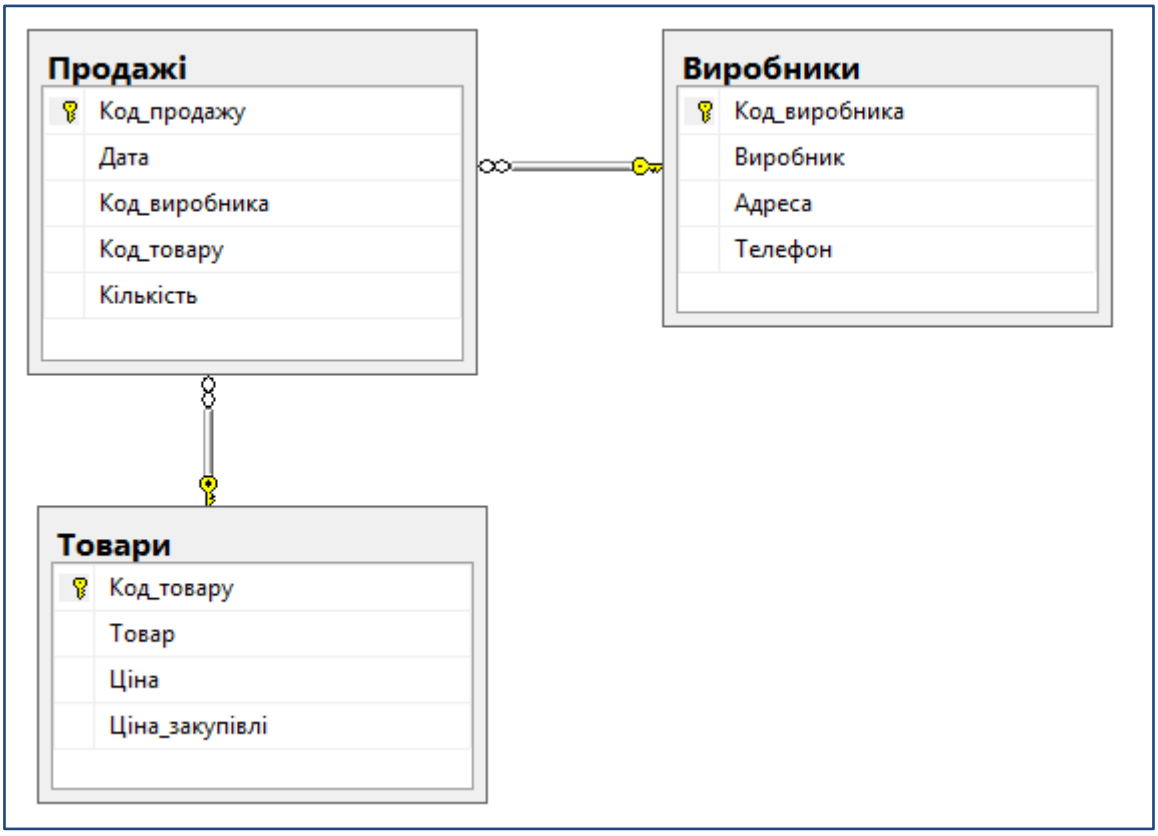


Рис. 1.4. **Зв'язки між таблицями в базі даних *Хліб***

Виконання

1. Розкрийте вузол бази даних у вікні **Server Explorer**.

2. Клацніть ПКМ на значку **Tables** бази даних і з контекстового меню виберіть команду **Add – New Table**.

3. Задайте схему таблиці ***Товари***. Імена полів та їхніх властивостей подано в табл. 1.1.

Таблиця 1.1

Поля таблиці *Товари*

№ п/п	Поле (Name)	Властивість	Значення
1	Код_товару	Data Type	int
		Identity Specification (Is Identity)	True
		Primary Key	True
2	Товар	Data Type	nvarchar
		Length	25
3	Ціна	Data Type	money
4	Ціна_закупівлі	Data Type	money

Для Visual Studio 2010:

4. Клацніть кнопку **Save** і введіть ім'я таблиці ***Товари***.

Для Visual Studio 2012 і вище:

5. Введіть ім'я таблиці ***Товари*** у першому рядку вкладки **T-SQL** (рис. 1.5).

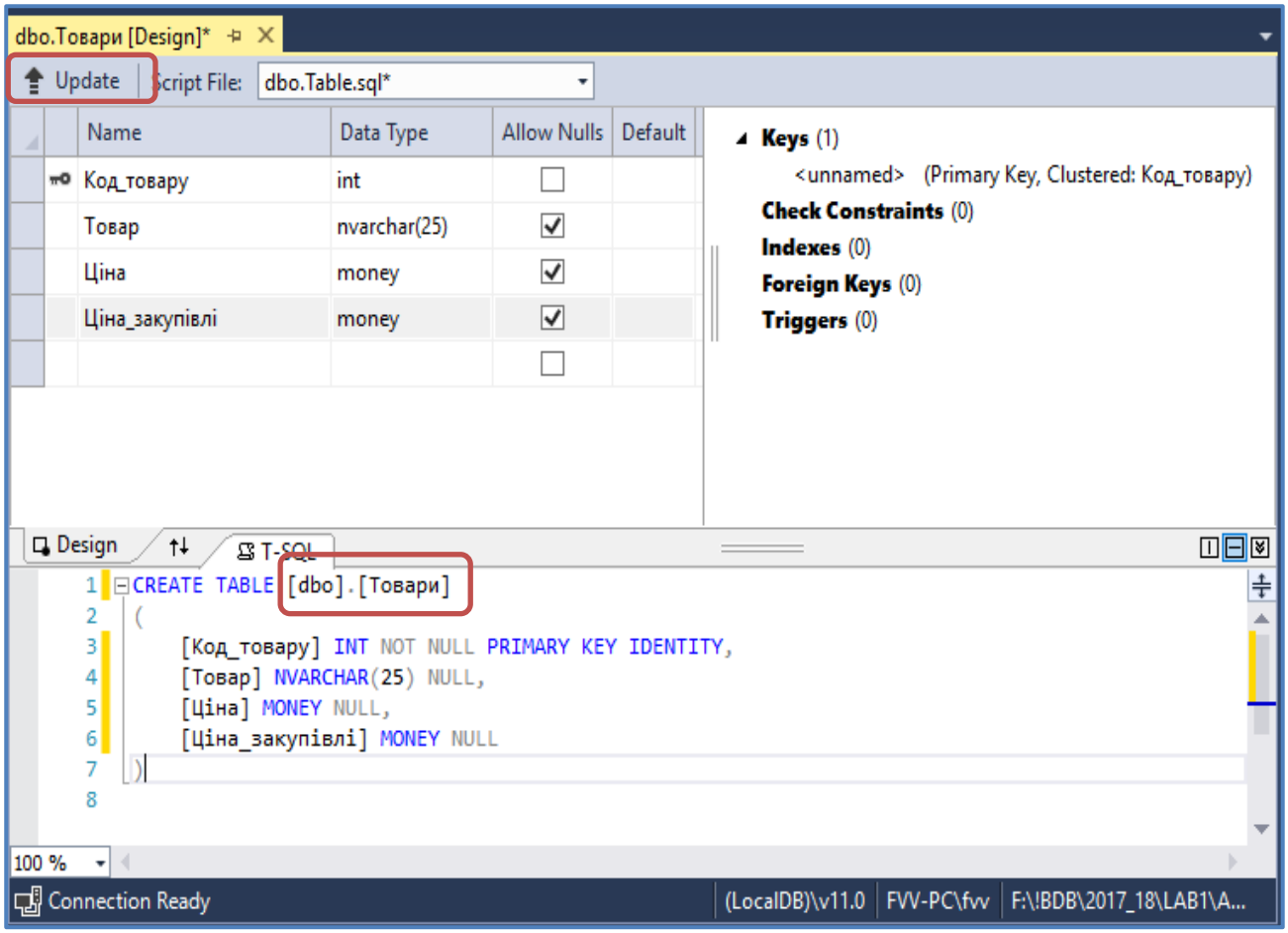


Рис. 1.5. **Вікно конструктора таблиці**

6. Клацніть кнопку **Update** у верхній частині вікна конструктора таблиці (див. рис. 1.5), а потім – кнопку **Update Database** у вікні, що з'явилося.

7. Обновіть вузол **Tables** у вікні **Server Explorer**, а потім розкрийте вузол таблиці ***Товари***. Перегляньте схему таблиці ***Товари*** у вікні **Server Explorer** (рис. 1.6).

8. У звіті з лабораторної роботи помістіть текст завдання і скриншоти, аналогічні рис. 1.5, 1.6.

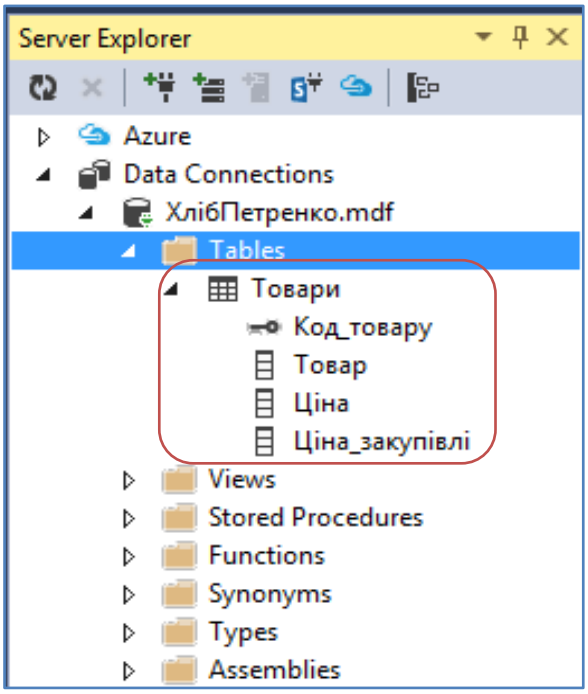


Рис. 1.6. **Схема таблиці *Товари* у вікні Server Explorer**

Завдання 2

Створіть таблицю ***Виробники*** у базі даних ***Хліб*** з використанням візуальних засобів.

Виконання

Побудова таблиці ***Виробники*** здійснюється аналогічно описаному у завданні 1. Імена полів та їхніх властивостей подано в табл. 1.2.

Таблиця 1.2

Поля таблиці *Виробники*

№ п/п	Поле (Name)	Властивість	Значення
1	Код_виробника	Data Type	int
		Identity Specification (Is Identity)	True
		Primary Key	True
2	Виробник	Data Type	nvarchar
		Length	20
3	Адреса	Data Type	nvarchar
		Length	30
4	Телефон	Data Type	nvarchar
		Length	15

Додайте відповідні дані до звіту з лабораторної роботи.

Завдання 3

Створіть таблицю ***Продажі*** у базі даних ***Хліб*** з використанням візуальних засобів.

Виконання

Побудова таблиці ***Продажі*** здійснюється аналогічно описаному у завданні 1. Імена полів та їхніх властивостей подано в табл. 1.3.

Таблиця 1.3

Поля таблиці *Продажі*

№ п/п	Поле (Name)	Властивість	Значення
1	Код_продажу	Data Type	int
		Identity Specification (Is Identity)	True
		Primary Key	True
2	Дата	Data Type	date
3	Код_виробника	Data Type	int
4	Код_товару	Data Type	int
5	Кількість	Data Type	smallint

Додайте відповідні дані до звіту з лабораторної роботи.

Завдання 4

Встановити зв'язки між таблицями в базі даних ***Хліб***.

Ідея розв'язку

У таблиці ***Продажі*** може бути кілька записів про продаж того самого товару від різних виробників протягом одного дня. Ще більше буде записів у таблиці ***Продажі***, у якій згадується той самий товар протягом кількох днів. Тому таблиці ***Товари*** та ***Продажі*** пов'язані відношенням один-до-багатьох.

З аналогічних причин таблиці ***Виробники*** та ***Продажі*** пов'язані відношенням один-до-багатьох.

Отже, у базі даних ***Хліб*** таблиці ***Товари*** та ***Виробники*** є батьківськими, а таблиця ***Продажі*** – дочірньою.

Зв'язок між батьківськими таблицями та дочірньою встановлюється за співпадінням значень первинного ключа у батьківській таблиці з відповідним зовнішнім ключем у дочірній. Наприклад, таблиці ***Товари***

та ***Продажі*** пов'язані первинним ключем ***Код_товару*** у таблиці ***Товари*** та однойменним зовнішнім ключем у таблиці ***Продажі***.

Установлення зв'язків між таблицями у різних версіях *Visual Studio* здійснюється різними засобами. У *Visual Studio 2010* є діаграми баз даних, а в подальших версіях вони відсутні. Тому у *Visual Studio 2010* для цього можна застосувати візуальні засоби, а в подальших версіях – засоби мови SQL.

Примітка: у всіх версіях Management Studio є діаграми баз даних.

Виконання

Для Visual Studio 2010:

1. Додайте діаграму до бази даних за допомогою контекстового меню значка **Diagrams**.

2. Додайте всі три таблиці до діаграми у вікні, що відкрилося.

3. Перетягніть первинний ключ ***Код_товару*** з батьківської таблиці ***Товари*** на однойменний зовнішній ключ дочірньої таблиці ***Продажі***.

4. Повторіть п. 3 для таблиць ***Виробники*** та ***Продажі***.

5. На рис. 1.7 подано схему бази даних ***Хліб***. У звіті з лабораторної роботи помістіть текст завдання і скриншот, аналогічний рис. 1.7.

Для Visual Studio 2012 і вище:

1. Клацніть ПКМ в конструкторі дочірньої таблиці ***Продажі*** на тексті **Foreign Keys**, що знаходиться у правій панелі вікна конструктора, та виберіть команду **Add New Foreign Key**.

2. Замініть в кінці вкладки **T-SQL** текст

```
CONSTRAINT [FK_Продажі_ToTable] FOREIGN KEY ([Column]) REFERENCES
[ToTable]([ToTableColumn])
```

на такий:

```
CONSTRAINT [FK_ТовариПродажі] FOREIGN KEY ([Код_товару]) REFERENCES
[dbo].[Товари] ([Код_товару]) ON DELETE CASCADE,
CONSTRAINT [FK_ВиробникиПродажі] FOREIGN KEY ([Код_виробника])
REFERENCES [dbo].[Виробники] ([Код_виробника]) ON DELETE CASCADE
```

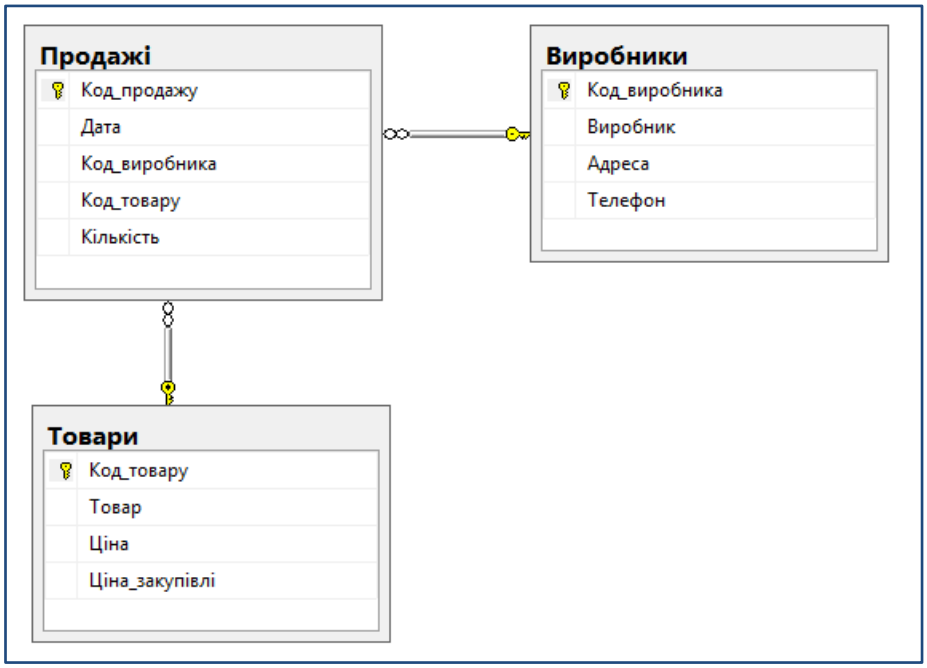


Рис. 1.7. **Схема бази даних *Хліб***

3. Клацніть кнопку **Update** у верхній частині вікна конструктора таблиці, а потім – кнопку **Update Database** у вікні, що з'явилося.

4. На рис. 1.8 подано схему дочірньої таблиці ***Продажі*** зі встановленими зв'язками.

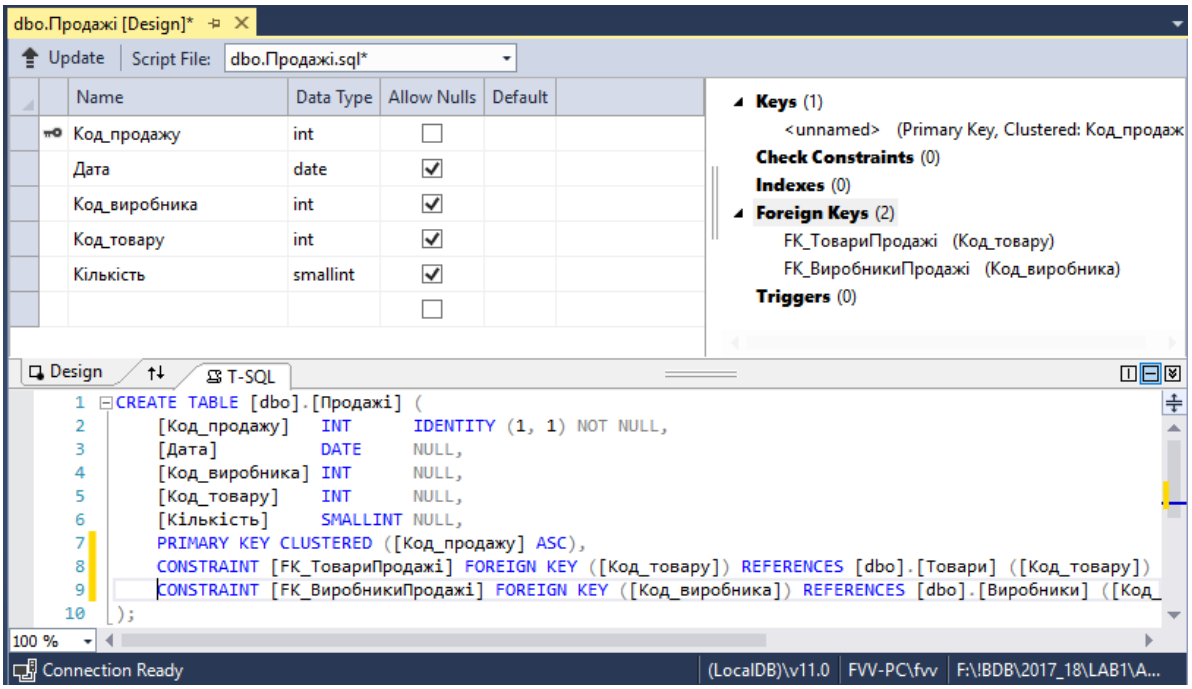


Рис. 1.8. **Схема таблиці *Продажі* зі встановленими зв'язками**

У звіті з лабораторної роботи помістіть текст завдання і скриншот, аналогічний рис. 1.8.

1.3. Заповнення таблиць даними з використанням візуальних засобів

Завдання

Заповнити даними таблиці бази даних ***Хліб***.

Ідеї розв'язку

Спочатку вводять дані до батьківських таблиць, а потім – до дочірніх, щоб не порушити посилальну цілісність даних. Ця цілісність передбачає, що значення зовнішніх ключів у дочірній таблиці можуть вибиратися тільки з множини значень ключа відповідної батьківської таблиці. Наприклад, можна продати тільки той товар, що є в таблиці ***Товари***, і від того виробника, який є в таблиці ***Виробники***.

Значення первинних ключів не вводять, оскільки вони мають властивість Identity, тобто автоматично задаються сервером під час збереження даних.

Введений запис автоматично зберігається в базі даних у разі переходу на інший рядок.

Значення первинних ключів відображаються після клацання кнопки **Refresh** у вікні даних таблиці.

Виконання

1. Заповніть даними таблицю ***Товари***.

Для цього:

1.1. розкрийте вузол бази даних у вікні **Server Explorer**;

1.2. клацніть ПКМ на значку таблиці ***Товари*** бази даних ***Хліб***, що знаходиться у вікні **Server Explorer**, і з контекстового меню виберіть команду **Show Table Data**;

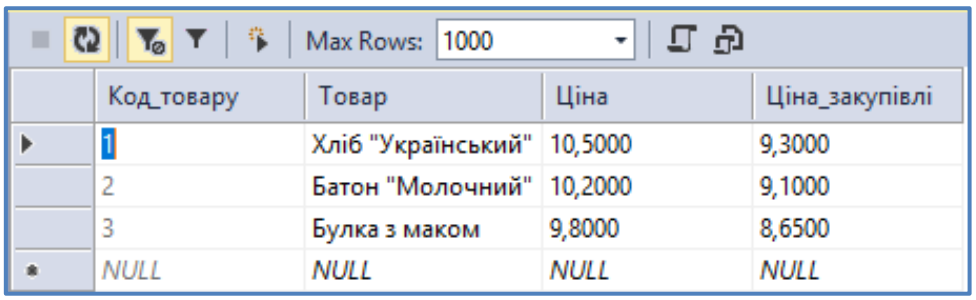
1.3. введіть дані, які подано в табл. 1.4.

Таблиця 1.4

Дані таблиці *Товари*

Товар	Ціна	Ціна_закупівлі
Хліб "Український"	10,50	9,30
Батон "Молочний"	10,20	9,10
Булка з маком	9,80	8,65

1.4. Обновіть зображення даних у таблиці бази даних, клацнувши кнопку **Refresh** у вікні даних. Таблицю з даними подано на рис. 1.9.



Код_товару	Товар	Ціна	Ціна_закупівлі
1	Хліб "Український"	10,5000	9,3000
2	Батон "Молочний"	10,2000	9,1000
3	Булка з маком	9,8000	8,6500
NULL	NULL	NULL	NULL

Рис. 1.9. **Дані таблиці *Товари***

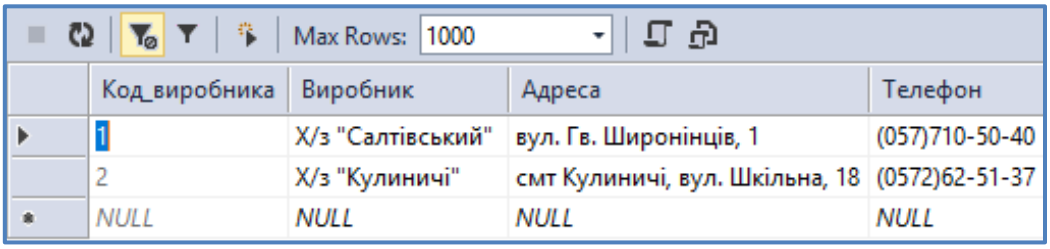
2. Заповніть даними таблицю ***Виробники*** аналогічно п. 1, дані якої подано в табл. 1.5.

Таблиця 1.5

Дані таблиці *Виробники*

Виробник	Адреса	Телефон
Х/з "Салтівський"	вул. Гв. Широнінців, 1	(057)710-50-40
Х/з "Кулиничі"	смт Кулиничі, вул. Шкільна, 18	(0572)62-51-37

Після оновлення дані таблиці ***Виробники*** подано на рис. 1.10.



Код_виробника	Виробник	Адреса	Телефон
1	Х/з "Салтівський"	вул. Гв. Широнінців, 1	(057)710-50-40
2	Х/з "Кулиничі"	смт Кулиничі, вул. Шкільна, 18	(0572)62-51-37
NULL	NULL	NULL	NULL

Рис. 1.10. **Дані таблиці *Виробники***

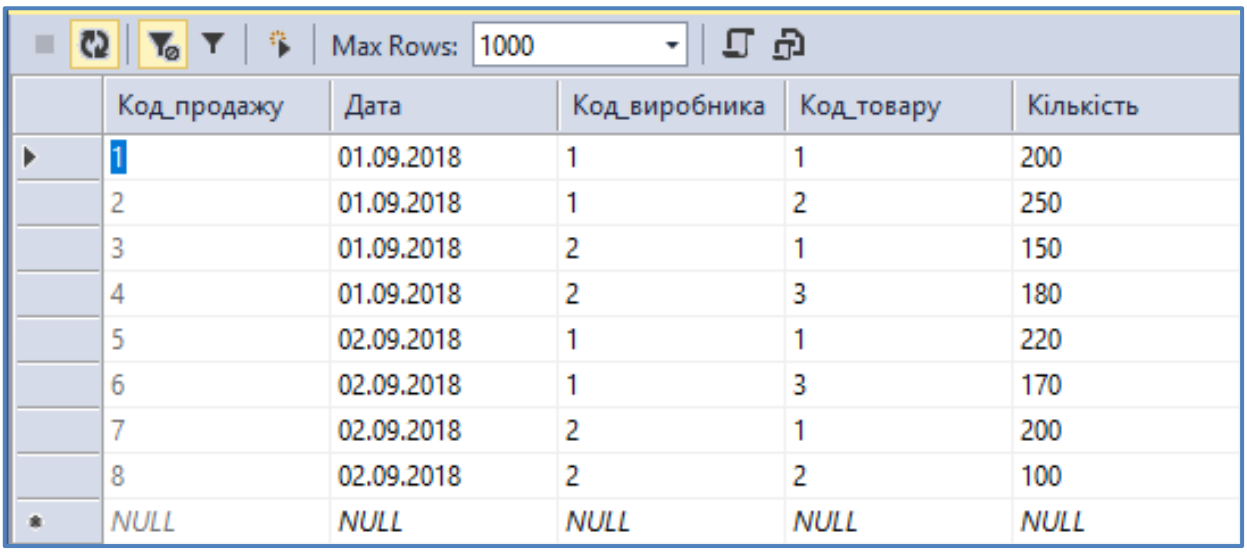
3. Заповніть даними таблицю ***Продажі*** аналогічно п. 1, дані якої подано в табл. 1.6.

Таблиця 1.6

Дані таблиці *Продажі*

Дата	Код_виробника	Код_товару	Кількість
01.09.2018	1	1	200
01.09.2018	1	2	250
01.09.2018	2	1	150
01.09.2018	2	3	180
02.09.2018	1	1	220
02.09.2018	1	3	170
02.09.2018	2	1	200
02.09.2018	2	2	100

Після оновлення дані таблиці ***Продажі*** подано на рис. 1.11.



Max Rows: 1000

Код_продажу	Дата	Код_виробника	Код_товару	Кількість
1	01.09.2018	1	1	200
2	01.09.2018	1	2	250
3	01.09.2018	2	1	150
4	01.09.2018	2	3	180
5	02.09.2018	1	1	220
6	02.09.2018	1	3	170
7	02.09.2018	2	1	200
8	02.09.2018	2	2	100
NULL	NULL	NULL	NULL	NULL

Рис. 1.11. **Дані таблиці *Продажі***

4. У звіті з лабораторної роботи помістіть текст завдання і скриншоти, аналогічні рис. 1.9 – 1.11.

1.4. Створення бази даних засобами мови DDL

Завдання

Створіть базу даних ***ТоргівляПрізвище*** в СУБД *MS SQL Server* у вигляді mdf-файла, використовуючи засоби мови DDL.

Примітка: замість слова ***Прізвище*** в значенні імені бази даних потрібно вставити своє прізвище. Тоді база даних матиме відповідне ім'я, наприклад ***Торгівля-Петренко***.

Ідея розв'язку

Запити мови DDL задають у вікні редактора T-SQL. Це вікно у різних версіях *Visual Studio* викликається по-різному. У *Visual Studio 2010* для цього використовують команду **Data – Transact-SQL Editor**, а в подальших версіях *Visual Studio* достатньо з контекстового меню значка бази даних вибрати команду **New Query**.

Виконання

1. Відкрийте вікно редактора T-SQL і введіть запит на створення бази даних у форматі

```
CREATE DATABASE i'мя_бази_даних
ON (NAME = логічне_і'мя_файла,FILENAME = шлях_і_фізичне_і'мя_файла)
```

наприклад,

```
CREATE DATABASE ТоргівляПетренко On (NAME = ТоргівляПетренко,
FILENAME='F:\fvv\DB\ТоргівляПетренко.mdf')
```

Примітка: у параметрі **FILENAME** слід указати повний шлях до mdf-файла, в якому зберігається база даних.

2. Запустіть запит на виконання, клацнувши кнопку **Execute** у вікні редактора.

3. Відкрийте вікно папки, що вказана у параметрі **FILENAME**; переконайтеся, що в ній з'явилося два файли – з розширенням mdf' (файл даних) і розширенням ldf (файл журналу).

4. Для спрощення подальших операцій скопіюйте нові файли в проект.

Для цього:

4.1. Клацніть ПКМ на значку проекту у вікні **Solution Explorer** і з контекстового меню виберіть команду **Add – Existing Item**.

4.2. Перейдіть у вікно папки з файлами нової бази даних, установіть у фільтрі значення **All files(*.*)**, виділіть mdf-файл нової бази даних і клацніть кнопку **Add**.

4.3. Зачекайте деякий час поки у вікнах **Solution Explorer** і **Server Explorer** з'являться значки нової бази даних ***Торгівля***.

5. Відкрийте вузол значка бази даних ***Торгівля*** у вікні **Server Explorer**.

6. У звіті з лабораторної роботи помістіть текст завдання, запит на створення бази даних ***Торгівля*** і скриншоти вікон **Solution Explorer** і **Server Explorer** із значками нової бази даних.

1.5. Створення таблиць засобами мови DDL

Постановка задачі

База даних ***Торгівля*** містить такі таблиці:

Замовники (Код_замовника, Прізвище_замовника, Місто, Рейтинг);

Продавці (Код_продавця, Прізвище_продавця, Місто, Комісійні);

Замовлення (Код_замовлення, Дата_замовлення, Сума, Код_продавця, Код_замовника).

Ідеї розв'язку

Після того як створено базу даних, для виконання подальших запитів з нею потрібно встановити з'єднання у вікні редактора T-SQL.

У *Visual Studio 2010* для цього виконують такі дії.

1. Клацають на значку бази даних у вікні **Server Explorer**.

2. Копіюють в буфер обміну значення властивості **Connection String** з вікна **Properties** клавішами **Ctrl-C**.

3. Виконують команду **Data – Transact-SQL Editor – New Query Connection**.

4. Клацають кнопку **Options** у вікні **Connect to Server**, а потім переходять на вкладку **Additional Connection Parameters**, вставляють з буфера рядок з'єднання клавішами **Ctrl-V** і клацають кнопку **Connect**.

5. Установлюють ім'я бази даних у полі зі списком **Database** вікна редактора.

У подальших версіях *Visual Studio* для того, щоб встановити з'єднання з базою даних, достатньо з контекстового меню значка бази даних вибрати команду **New Query**.

Завдання 1

Створіть таблицю ***Замовники*** у базі даних ***Торгівля*** з використанням мови DDL.

Виконання

1. Відкрийте вікно редактора T-SQL, що з'єднане з базою даних ***Торгівля***, і введіть запит на створення таблиці ***Замовники***

```
CREATE TABLE Замовники
    (Код_замовника int IDENTITY (1, 1) Primary Key,
    Прізвище_замовника NVARCHAR(30),
    Місто NVARCHAR(30),
    Рейтинг INT
     )
```

2. Запустіть запит на виконання, клацнувши у вікні редактора кнопку **Execute**.

3. Обновіть вузол **Tables** у вікні **Server Explorer**, а потім розкрийте вузол таблиці ***Замовники***. Перегляньте схему таблиці ***Замовники*** у вікні **Server Explorer** (рис. 1.12).

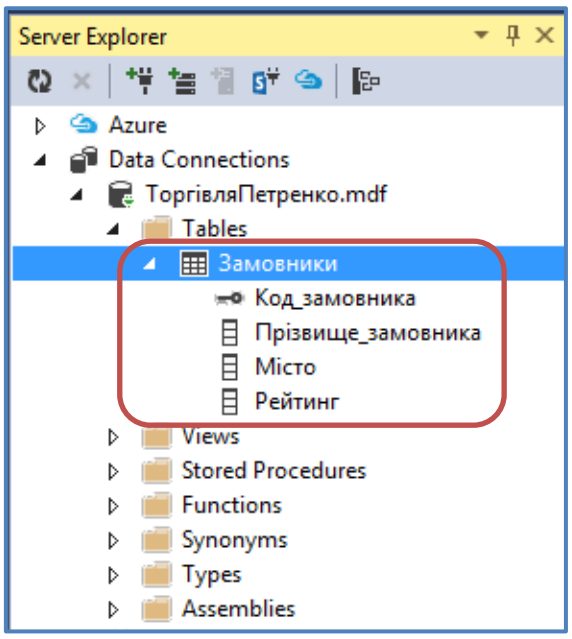


Рис. 1.12. **Схема таблиці *Замовники* у вікні Server Explorer**

4. У звіті з лабораторної роботи помістіть текст завдання, запит на створення таблиці ***Замовники*** та скриншот, аналогічний рис. 1.12.

Завдання 2

Створіть таблицю ***Продавці*** у базі даних ***Торгівля*** з використанням мови DDL.

Виконання

Побудова таблиці ***Продавці*** здійснюється аналогічно описаному в завданні 1. Запит на створення таблиці ***Продавці*** має такий вигляд:

```
CREATE TABLE Продавці
    (Код_продавця int IDENTITY (1, 1) Primary Key,
    Прізвище_продавця NVARCHAR(30),
    Місто NVARCHAR(30),
    Комісійні DECIMAL(5,2)
     )
```

Додайте відповідні дані до звіту з лабораторної роботи.

Завдання 3

Створіть таблицю ***Замовлення*** у базі даних ***Торгівля*** з використанням візуальних засобів. Під час створення таблиці встановіть зв'язки з батьківськими таблицями ***Замовники*** та ***Продавці***, використавши зовнішні ключі.

Виконання

Побудова таблиці ***Замовлення*** здійснюється аналогічно описаному у завданні 1.

Запит на створення таблиці ***Замовлення*** має такий вигляд:

```
CREATE TABLE Замовлення
    (Код_замовлення INT IDENTITY (1, 1) Primary Key,
     Дата_замовлення DATE,
     Сума MONEY,
     Код_продавця INT FOREIGN KEY REFERENCES Продавці(Код_продавця),
     Код_замовника INT FOREIGN KEY REFERENCES Замовники(Код_замовника)
     )
```

Додайте відповідні дані до звіту з лабораторної роботи.

1.6. Заповнення таблиць даними засобами мови SQL

Завдання

Заповніть даними таблиці бази даних ***Торгівля*** з використанням команд SQL. В результаті таблиці повинні зберігати такі кількості записів:

таблиця ***Замовники*** – не менше 10 записів;

таблиця ***Продавці*** – не менше 10 записів;

таблиця ***Замовлення*** – не менше 30 записів.

Ідеї розв'язку

Спочатку вводять дані до батьківських таблиць, а потім – до дочірніх, щоб не порушити посилальну цілісність даних. Для цього використовують оператор INSERT. У найпростішому випадку для введення одного рядка таблиці він має такий формат:

```
INSERT INTO і'мя_таблиці VALUES (значення1, значення2,…)
```

Цей формат застосовують, якщо стовпці таблиці вказані в повному складі та в тому порядку, в якому вони подавались під час створення таблиці. Значення первинних ключів не вводять, оскільки вони мають властивість Identity, тобто автоматично задаються сервером під час збереження даних.

Якщо у разі наявності властивості Identity у первинного ключа потрібно самостійно задавати його значення, слід виконувати дві умови:

1) вказувати імена стовпців, значення яких вводять;

2) перед введенням таких даних виконувати для таблиці оператор SET IDENTITY_INSERT *і'мя_таблиці* ON.

Тобто оператор INSERT у цьому разі має такий формат:

```
INSERT INTO і'мя_таблиці (стовпець1, стовпець2, …)
VALUES (значення1, значення2,…)
```

Щоб текстові значення вводилися як символи Юнікоду, перед кожним з них потрібно ставити літеру N. У протилежному разі під час перегляду даних можуть виводитися символи **?**.

Наприклад, введення даних про першого замовника реалізується так:

```
SET IDENTITY_INSERT Замовники ON;
INSERT INTO Замовники (Код_замовника, Прізвище_замовника, Місто, Рейтинг) VALUES(1, N'Іваненко', N'Київ', 95);
SET IDENTITY_INSERT Замовники OFF;
```

Щоб дата оброблялася в вигляді 'ДД.ММ.РРРР' для SQL Server доцільно відразу ввести команду

```
SET LANGUAGE RUSSIAN
```

Наприклад, введення даних про замовлення від першого замовника у першого продавця реалізується так:

```
SET LANGUAGE RUSSIAN;
SET IDENTITY_INSERT Замовлення ON;
INSERT INTO Замовлення (Код_замовлення, Дата_замовлення, Сума,
Код_продавця, Код_замовника) VALUES(1, '01.09.2018', 2000, 1, 1);
SET IDENTITY_INSERT Замовлення OFF;
```

Виконання

1. Заповніть даними таблицю ***Замовники***.

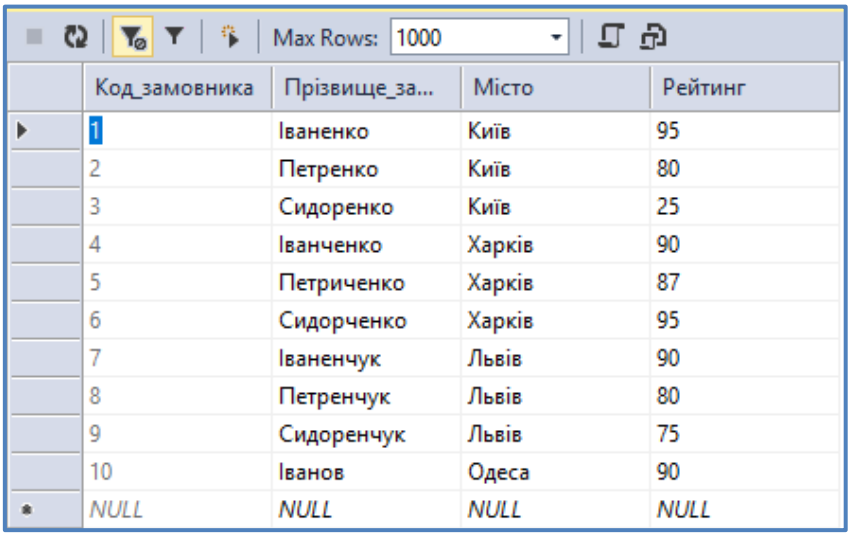
Для цього:

1.1. Введіть у вікні редактора T-SQL текст такого скрипту:

```
SET IDENTITY_INSERT Замовники ON
INSERT INTO Замовники (Код_замовника, Прізвище_замовника, Місто,
Рейтинг) VALUES(1, N'Іваненко', N'Київ', 95);
INSERT INTO Замовники (Код_замовника, Прізвище_замовника, Місто,
Рейтинг) VALUES(2, N'Петренко', N'Київ', 80);
INSERT INTO Замовники (Код_замовника, Прізвище_замовника, Місто,
Рейтинг) VALUES(3, N'Сидоренко',N'Київ', 25);
INSERT INTO Замовники (Код_замовника, Прізвище_замовника, Місто,
Рейтинг) VALUES(4, N'Іванченко',N'Харків', 90);
INSERT INTO Замовники (Код_замовника, Прізвище_замовника, Місто,
Рейтинг) VALUES(5, N'Петриченко',N'Харків', 87);
INSERT INTO Замовники (Код_замовника, Прізвище_замовника, Місто,
Рейтинг) VALUES(6, N'Сидорченко',N'Харків', 95);
INSERT INTO Замовники (Код_замовника, Прізвище_замовника, Місто,
Рейтинг) VALUES(7, N'Іваненчук',N'Львів', 90);
INSERT INTO Замовники (Код_замовника, Прізвище_замовника, Місто,
Рейтинг) VALUES(8, N'Петренчук',N'Львів', 80);
INSERT INTO Замовники (Код_замовника, Прізвище_замовника, Місто,
Рейтинг) VALUES(9, N'Сидоренчук',N'Львів', 75);
INSERT INTO Замовники (Код_замовника, Прізвище_замовника, Місто,
Рейтинг) VALUES(10, N'Іванов',N'Одеса', 90);
SET IDENTITY_INSERT Замовники OFF
```

1.2. Запустіть запит на виконання, клацнувши у вікні редактора кнопку **Execute**.

1.3. Обновіть вузол **Tables** у вікні **Server Explorer**, а потім з контекстового меню таблиці ***Замовники*** виберіть команду **Show Table Data**. Перегляньте дані таблиці ***Замовники*** у вікні даних (рис. 1.13).



Код_замовника	Прізвище_за...	Місто	Рейтинг
1	Іваненко	Київ	95
2	Петренко	Київ	80
3	Сидоренко	Київ	25
4	Іванченко	Харків	90
5	Петриченко	Харків	87
6	Сидорченко	Харків	95
7	Іваненчук	Львів	90
8	Петренчук	Львів	80
9	Сидоренчук	Львів	75
10	Іванов	Одеса	90
NULL	NULL	NULL	NULL

Рис. 1.13. **Дані таблиці *Замовники***

1.4. У звіті з лабораторної роботи помістіть текст завдання, скрипт на заповнення таблиці ***Замовники*** та скриншот, аналогічний рис. 1.13.

2. Заповніть таблицю ***Продавці*** у базі даних ***Торгівля*** з використанням мови DDL (не менше десяти записів), аналогічно описаному у завданні 1. Додайте відповідні дані до звіту з лабораторної роботи.

3. Заповніть таблицю ***Замовлення*** у базі даних ***Торгівля*** з використанням мови DDL (не менше тридцяти записів). Під час заповнення таблиці використовуйте значення зовнішніх ключів, які співпадають зі значеннями первинних ключів батьківських таблиць ***Замовники*** та ***Продавці***. Додайте відповідні дані до звіту з лабораторної роботи.

1.7. Створення індивідуальної бази даних. Створення і заповнення таблиць засобами мови SQL

Завдання

За вибраним варіантом створіть базу даних і таблиці в ній; установіть зв'язки між таблицями та заповніть даними таблиці, використавши засоби мови SQL.

Варіанти баз даних

1. База даних **Готель** містить такі таблиці:

Клієнти (Код клієнта, Прізвище, Ім'я, По батькові, Паспортні дані, Коментар);

Номери (Код номера, Номер, Кількість місць, Комфортність, Ціна);

Поселення (Код поселення, Код клієнта, Код номера, Дата поселення, Дата звільнення, Примітка).

2. База даних **Ломбард** містить такі таблиці:

Клієнти (Код клієнта, Прізвище, Ім'я, По батькові, Номер паспорта, Серія паспорта, Дата видачі паспорта);

Категорії товарів (Код категорії товарів, Назва, Примітка);

Здача в ломбард (Код, Код категорії товарів, Код клієнта, Опис товару, Дата здачі, Дата повернення, Сума, Комісійні).

3. База даних **Оптово-роздрібна компанія** містить такі таблиці:

Товари (Код товару, Найменування, Оптова ціна, Роздрібна ціна, Опис);

Покупці (Код покупця, Телефон, Контактна особа, Адреса);

Угоди (Код угоди, Дата угоди, Код товару, Кількість, Код покупця, Ознака оптового).

4. База даних **Оптова компанія** містить такі таблиці:

Товари (Код товару, Ціна, Доставка, Опис);

Замовники (Код замовника, Найменування, Адреса, Телефон, Контактна особа);

Замовлення (Код замовлення, Код замовника, Код товару, Кількість, Дата).

5. База даних **Бюро з працевлаштування** містить такі таблиці:

Роботодавці (Код роботодавця, Назва, Вид діяльності, Адреса, Телефон);

Здобувачі (Код здобувача, Прізвище, Ім'я, По батькові, Кваліфікація, Вид діяльності, Інші дані, Передбачуваний розмір заробітної плати);

Угоди (Код здобувача, Код роботодавця, Посада, Комісійні).

6. База даних **Нотаріальна контора** містить такі таблиці:

Клієнти (Код клієнта, Назва, Вид діяльності, Адреса, Телефон);

Угоди (Код угоди, Код клієнта, Код послуги, Сума, Комісійні, Опис);

Послуги (Код послуги, Назва, Опис).

7. База даних **Страхова компанія** містить такі таблиці:

Договори (Номер договору, Дата висновку, Страхова сума, Тарифна ставка, Код філії, Код виду страхування);

Вид страхування (Код виду страхування, Найменування);

Філія (Код філії, Найменування філії, Адреса, Телефон).

8. База даних **Запасні частини** містить такі таблиці:

Постачальники (Код постачальника, Назва, Адреса, Телефон);

Деталі (Код деталі, Назва, Артикул, Ціна, Примітка);

Поставки (Код постачальника, Код деталі, Кількість, Дата).

9. База даних **Навчальний заклад** містить такі таблиці:

Групи (Номер групи, Спеціальність, Відділення, Кількість студентів);

Викладачі (Код викладача, Прізвище, Ім'я, По батькові, Телефон, Стаж);

Навантаження (Код викладача, Номер групи, Кількість годин, Предмет, Тип заняття, Оплата).

10. База даних **Університет** містить такі таблиці:

Студенти (Код студента, Прізвище, Ім'я, По батькові, Адреса, Телефон);

Предмети (Код предмета, Назва, Обсяг лекцій, Обсяг практик, Обсяг лабораторних робіт);

Екзамени (Код студента, Код предмета, Оцінка).

11. База даних **Академія** містить такі таблиці:

Викладачі (Код викладача, Прізвище, Ім'я, По батькові, Учений ступінь, Посада, Стаж);

Предмети (Код предмета, Назва, Кількість годин);

Навантаження (Код викладача, Код предмета, Номер групи).

12. База даних **Комерційна компанія** містить такі таблиці:

Співробітники (Код співробітника, Прізвище, Ім'я, По батькові, Оклад);

Види робіт (Код виду, Опис, Оплата за день);

Роботи (Код співробітника, Код виду, Дата початку, Дата закінчення).

13. База даних **Ремонт верстатів** містить такі таблиці:

Види верстатів (Код виду верстата, Країна, Рік випуску, Марка);

Види ремонту (Код ремонту, Назва, Тривалість, Вартість, Примітки).

Ремонт (Код виду верстата, Код ремонту, Дата початку, Примітки).

14. База даних **Туристична компанія** містить такі таблиці:

Маршрути (Код маршруту, Країна, Клімат, Тривалість, Готель, Вартість);

Путівки (Код маршруту, Код клієнта, Дата відправлення, Кількість, Знижка);

Клієнти (Код клієнта, Прізвище, Ім'я, По батькові, Адреса, Телефон).

15. База даних **Перевезення вантажів** містить такі таблиці:

Маршрути (Код маршруту, Назва, Довжина, Кількість днів у дорозі, Оплата);

Водії (Код водія, Прізвище, Ім'я, По батькові, Стаж);

Виконана робота (Код маршруту, Код водія, Дата відправлення, Дата повернення, Премія).

16. База даних **Телефонна компанія** містить такі таблиці:

Абоненти (Код абонента, Номер телефону, Адреса);

Міста (Код міста, Назва, Тариф денний, Тариф нічний);

Переговори (Код переговорів, Код абонента, Код міста, Дата, Кількість хвилин, Час доби).

17. База даних **Банк** містить такі таблиці:

Види кредитів (Код виду, Назва, Умови отримання, Ставка, Строк);

Клієнти (Код клієнта, Назва, Вид власності, Адреса, Телефон, Контактна особа);

Кредити (Код виду, Код клієнта, Сума, Дата видачі).

18. База даних **Бухгалтерія** містить такі таблиці:

Відділи (Код відділу, Назва, Кількість співробітників);

Види витрат (Код виду, Назва, Опис, Гранична норма);

Витрати (Код витрати, Код виду, Код відділу, Сума, Дата).

19. База даних **Прокат книг** містить такі таблиці:

Книги (Код книги, Назва, Автор, Станова вартість, Вартість прокату, Жанр);

Читачі (Код читача, Прізвище, Ім'я, По батькові, Адреса, Телефон);

Видані книги (Код книги, Код читача, Дата видачі, Дата повернення).

20. База даних **Прокат автомобілів** містить такі таблиці:

Автомобілі (Код автомобіля, Марка, Вартість, Вартість прокату, Тип).

Клієнти (Код клієнта, Прізвище, Ім'я, По батькові, Адреса, Телефон);

Видані автомобілі (Код автомобіля, Код клієнта, Дата видачі, Дата повернення).

21. База даних **Зустрічі з товаришами** містить такі таблиці:

Товариші (Код товариша, Ім'я, Вік);

Теми (Код теми, Назва, Опис);

Зустрічі (Код зустрічі, Дата, Код товариша, Код теми, Оцінка).

Виконання

Повторіть дії, аналогічні описаним у пп. 4 – 6.

У звіті з лабораторної роботи помістіть формулювання вибраного варіанту, тексти завдань, запитів на їх виконання і скриншоти з результатами.

Завдання для самостійного виконання

1. Додайте до бази даних ***Хліб*** ще дві таблиці:

Накладні(Код_накладної, Номер_накладної, Дата, Код_виробника);

Товари_накладних (Код_товару_накладної, Код_накладної, Код_товару, Кількість).

Установіть зв'язки між цими та іншими таблицями бази даних.

У нові таблиці запишіть дані про товари, що надходять до хлібного кіоску.

2. Для кожної з наступних задач зробіть копію з папкою проекту та реалізуйте:

2.1. зміну схеми бази даних, якщо магазин фірмовий, тобто продає товари тільки свого виробника (хлібозаводу);

2.2. умову, щоб у базу даних записувалася інформація про продажі, що надходить з декількох магазинів, які утворюють мережу;

2.3. зміну схеми, щоб передбачити продаж декількох видів хліба "Український", "Родзинка", "Бородинський", батонів "Молочний", "Київський", "Нарізний" тощо. Тобто товари треба розподілити за групами, наприклад хліб, батони, булки;

2.4. щоб для визначення в подальшому прибутку кіоску, зберігати в базі даних закупівельну ціну товарів кожного виду. У різних виробників вона різна.

3. До трьох таблиць, що вибрані за варіантом у завданні 1.7, додайте ще дві таблиці, які пов'язані між собою відношенням один-до-багатьох. Ці дві таблиці мають відображати один документ, наприклад, як у п. 1 документ *Накладна*. Створіть ці таблиці, встановіть зв'язки з іншими та заповніть їх даними.

Примітка. У звіті за кожним завданням подайте текст завдання, запит на його виконання та скриншот результату.

Лабораторна робота 2
Побудова DML-запитів

Цілі роботи

1. Набуття практичних навичок побудови запитів на отримання даних таких типів: детальні, групування, підзапити, модифікування даних.

2. Створення і використання подань.

Перед виконанням роботи студент повинен знати:

основи використання *Visual Studio*;
основні об'єкти бази даних *MS SQL Server*;
основні команди мови SQL.

Після виконання лабораторної роботи студент повинен уміти:

самостійно будувати детальні запити, запити, що містять групування даних за співпаданням значень вказаних полів, а також запити, до складу яких входять підзапити, запити на модифікування даних;

самостійно створювати подання, як об'єкти бази даних.

Підготовча частина

Хід роботи

2.1. Побудова базових запитів.

2.2. Створення запитів з групуванням даних.

2.3. Побудова запитів з підзапитами та запитів на зміну даних.

2.4. Створення і використання подань.

Форма звітності

За результатами виконання роботи оформити електронний звіт засобами Word.

У звіті за кожним завданням подати текст завдання, запит на його виконання та скриншот результату.

Усі скриншоти з виконаними запитами для SQL Server повинні мати коментар у першому рядку з прізвищем студента.

Коментар задається рядком з двома початковими мінусами:

```
-- Запит 2.1 Іванов В. І. 2 курс 1 група
SELECT * FROM Замовлення;
```

Критерії оцінювання

У 12-бальній системі оцінка результату захисту лабораторної роботи формується за такими правилами:

1. За кожне завдання з діапазону 2.1 – 2.4 може бути виставлено від 0 до 1,5 бала.

2. За кожне завдання з діапазону 1.1 – 1.5 із п. "Завдання для самостійного виконання" може бути виставлено від 0 до 1 бала, а за аналогічні завдання з індивідуальною базою даних від 0 до 1,5 бала.

3. За кілька варіантів розв'язання одного із завдань додається 1 бал.

4. За вибір варіанту, який з кількох варіантів вирішення є оптимальним, та обґрунтування вибору додається 1 бал.

5. За побудову запитів в іншій СУБД (Oracle, DB2 тощо) додається 1 бал за кожний запит.

6. За запізнення із захистом лабораторної роботи знімається 2 бали за кожний тиждень.

Отримана кількість балів з відповідей на кожне завдання лабораторної роботи підсумовується. У результаті такого підрахунку студент може отримати від 0 до 12 балів.

Рекомендована література: [1; 2; 6; 8; 12; 13; 16].

Основні поняття

Мова DML містить оператори, що дозволяють вибирати, додавати, видаляти та модифікувати дані, що зберігаються в таблицях. До її складу входять такі оператори:

SELECT – вибір даних з таблиць;

INSERT – додавання рядків даних до таблиці;

DELETE – вилучення рядків даних з таблиць;

UPDATE – зміна даних у таблицях.

Синтаксис оператора SELECT має такий вигляд:

```
SELECT [ALL | DISTINCT ] {*|[ім'я_стовпця [AS нове_ім'я]]} [,...n]
FROM ім'я_таблиці [[AS] псевдонім] [,...n]
[WHERE <умова_пошуку>]
[GROUP BY ім'я_стовпця [,...n]]
[HAVING <критерії вибору груп>]
[ORDER BY ім'я_стовпця [ASC | DESC] [,...n]]
```

Приклад 1. *Вибір усіх стовпців*. Отримайте повну інформацію про виробників.

```
SELECT *
FROM Співробітники;
```

Приклад 2. *Вибір окремих стовпців*. Для товарів отримайте інформацію про назву та ціну.

```
SELECT Товар, Ціна
FROM Товари;
```

Приклад 3. *Вибір рядків без повторень*. Отримайте значення кодів посади з таблиці Співробітники без повторювань.

```
SELECT DISTINCT Код_посади
FROM Співробітники;
```

Приклад 4. *Вибір стовпців з перейменуванням.* Отримайте відомості про товари, причому дати стовпцям нові імена: Товар → Назва товару, Ціна → Ціна продажу.

```
SELECT Товар AS "Назва товару", Ціна AS "Ціна продажу
FROM Товари;
```

Приклад 5. *Обчислювані стовпці у запиті*. Отримайте інформацію про товари, із зазначенням націнки (різниці між ціною продажу та закупівлі).

```
SELECT Товар, Ціна – Ціна_закупівлі AS Націнка
FROM Товари
```

Приклад 6. *Використання символьних рядків у запитах*. Отримайте дані про ціну продуктів, указавши позначення національної валюти.

```
SELECT Товар, Ціна, 'грн.' AS Валюта
FROM Товари;
```

Приклад 7. *Умовні вирази з використанням порівняння*. Отримайте перелік товарів, ціна яких перевищує 5 грн.

```
SELECT Товар, Ціна
FROM Товари
WHERE (Ціна>5);
```

Приклад 8. *Умовні вирази з використанням діапазону*. Отримайте перелік номерів накладних, за якими були поставлені товари 25 і 26 січня 2020 року.

```
SELECT Код_накладної
FROM Накладні
WHERE (Дата='25/01/2020' OR Дата='26/01/2020')
```

Приклад 9. *Умовні вирази з використанням приналежності до множини*. Визначте назви підприємств-виробників, коди яких мають значення 2, 4 або 5.

```
SELECT Виробник
FROM Виробники
WHERE (Код_виробника IN (2, 4, 5));
```

Приклад 10. *Умовні вирази з використанням відповідності шаблону*. Визначте товари, назви яких починаються на "ке".

```
SELECT Товар
FROM Товари
WHERE ( Товар LIKE 'ке%')
```

Приклад 11. *Умовні вирази з використанням відповідності шаблону*. Визначте товари, у назві яких зустрічається підрядок "повидло".

```
SELECT Товар
FROM Товари
WHERE ( Товар LIKE '%повидло%')
```

Приклад 12. *Умовні вирази з використанням значення NULL.* Визначте виробників, які не мають своїх сайтів у Інтернеті.

```
SELECT Виробник, Web_сайт
FROM Виробники
WHERE (Web_сайт IS NULL);
```

Приклад 13. *Упорядкування результатів вибірки даних (сортування)*. Отримайте повну інформацію про співробітників. Результат повинен бути впорядкований за спаданням дати народження співробітників.

```
SELECT *
FROM Співробітники
ORDER BY Прізвище_ІБ;
```

Приклад 14. *З'єднання з використанням речення WHERE*. Отримайте перелік товарів, які будь-коли реалізовувалися через кіоск.

```
SELECT DISTINCT Товар
FROM Товари, Продажі
WHERE (Товари.Код_товару = Продажі.Код_товару)
```

Приклад 15. *З'єднання таблиць з використанням JOIN*. Отримайте перелік товарів, які будь-коли реалізовувалися через кіоск (див. приклад 14). Результат отримати з використанням конструкції JOIN.

```
SELECT DISTINCT Товар
FROM Товари JOIN Продажі ON
    Товари.Код_товару = Продажі.Код_товару
```

Приклад 16. *Зовнішні з'єднання*. Визначте виробників, які ще не постачають у кіоск товарів.

```
SELECT Виробник
FROM Виробники LEFT JOIN Накладні ON
    Виробники.Код_виробника=Накладні.Код_виробника
WHERE (Номер_накладної IS NULL)
```

Приклад 17. *Використання псевдонімів у вибірці даних з кількох таблиць*. Визначте попарно всі товари, які належать до однієї групи.

Запит буде містити з'єднання таблиці Товари з самою собою.

```
SELECT DISTINCT T1.Товар, T2.Товар
FROM Товари T1, Товари T2
WHERE (T1.Код_групи=T2.Код_групи)
```

Приклад 18. *З'єднання за умовами, відмінними від рівності*. Знайдіть товари першого сорту, ціна яких перевищує ціну будь-якого товару вищого сорту.

```
SELECT DISTINCT T1.Товар
FROM Товари T1, Товари T2
WHERE (T1.Ціна>T2.Ціна AND T1.Сорт=1 AND T2.Сорт=0)
```

Приклад 19. *Операція декартового добутку*. Отримайте всі можливі комбінації назв товарів і виробників.

```
SELECT Товар, Виробник
FROM Товари, Виробники;
```

Користувачеві доступні такі основні агрегатні функції:

COUNT – визначає кількість записів у вихідному наборі SQL-запиту;

MIN – визначає найменше значення з множини значень у деякому стовпці запиту;

MAX – визначає найбільше значення з множини значень у деякому стовпці запиту;

SUM – обчислює суму множини значень, що містяться в певному стовпці відібраних запитом рядків;

AVG – визначає середнє арифметичне значення множини значень, що зберігаються у певному стовпці відібраних запитом рядків, тобто суму значень, що поділена на їхню кількість.

Приклад 20. *Використання агрегатних функцій за повним вихідним набором SQL-запиту*. Обчисліть загальну кількість підприємств-виробників.

```
SELECT COUNT(Код_виробника) AS "Кількість виробників"
FROM Виробники
```

Приклад 21. *Використання агрегатних функцій за повним вихідним набором SQL-запиту.* Знайдіть мінімальну, максимальну та середню ціну всіх товарів, що випускаються.

```
SELECT MIN(Ціна) AS "Ціна мінімальна",
    MAX(Ціна) AS "Ціна максимальна",
    AVG(Ціна) AS "Ціна середня"
FROM Товари
```

Приклад 22. *Групування даних*. Визначте загальний обсяг реалізації продукції на кожну дату січня.

```
SELECT Дата, SUM (Кількість*Ціна) AS "Обсяг реалізації"
FROM Продажі П INNER JOIN Товари Т ON П.Код_товару=Т.Код_товару
WHERE (Month(Дата)=1)
GROUP BY Дата
```

Приклад 23. *Групування даних*. Визначте мінімальну, максимальну та середню ціну товарів, що постачаються кожним підприємством-виробником.

```
SELECT Виробник,
    MIN(Ціна) AS Мінімальна,
    MAX(Ціна) AS Максимальна,
    AVG(Ціна) AS Середня
FROM Виробники В, Товари Т, Накладні Н, ТовариНакладних ТН
WHERE Т.Код_товару=ТН.Код_товару AND
    ТН.Номер_накладної=Н.Номер_накладної AND
```

```
    H.Код_виробника=B.Код_виробника
GROUP BY Виробник
```

Приклад 24. *Умова відбору груп*. Визначте, у які дні січня загальний обсяг реалізації продукції перевищував 200 грн.

```
SELECT Дата, SUM (Кількість*Ціна) AS "Обсяг реалізації"
FROM Продажі П INNER JOIN Товари Т ON П.Код_товару=Т.Код_товару
WHERE (Month(Дата)=1)
GROUP BY Дата
HAVING (SUM (Кількість*Ціна) > 300)
```

Додавання даних у таблицю здійснюється командою INSERT. Вона має такий формат:

```
INSERT INTO <Ім'я _таблиці > [(ім'я_стовпця [,...n])]
{VALUES (значення[,...n])|
<SELECT_оператор>}
```

Приклад 25. *Додавання даних*. Додайте дані про новий товар у таблицю Товари.

```
INSERT INTO Товари (Товар, Ціна, Ціна_закупівлі)
    VALUES (N'Хліб "Житній"', 12.50, 10.32 )
```

Оновлення даних здійснюється командою UPDATE. Вона має такий формат:

```
UPDATE Ім'я_таблиці
SET ім'я_стовпця=значення [,ім'я_стовпця=значення]
[WHERE умова_пошуку]
```

Приклад 26. *Оновлення даних*. У зв'язку з появою офіційного сайта Олексіївського хлібозаводу www.oleksiivka.kh.ua виконайте необхідну модифікацію у базі даних.

```
UPDATE Виробники
SET Web_сайт='www.oleksiivka.kh.ua'
WHERE (Виробник='Олексіївський х/з')
```

Видалення даних здійснюється командою DELETE. Вона має такий формат:

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

Приклад 27. *Видалення даних*. Видаліть з таблиці Товари відомості про продукцію третього сорту у зв'язку з тим, що вони більше не будуть реалізовуватися через торгову точку.

```
DELETE
FROM Товари
WHERE (Сорт=3)
```

Підзапити повертають свої результати у зовнішню команду, тобто результат виконання зовнішньої команди визначається результатом виконання внутрішнього запиту.

Некорельована форма підзапиту реалізується за правилом "зсередини-назовні", тобто зовнішній запит виконує свої дії, базуючись на результаті внутрішнього підзапиту.

Корельований підзапит працює дещо інакше. Зовнішня команда визначає умови для внутрішньої, після чого результати підзапиту повертаються у зовнішню команду.

Корельовані та некорельовані підзапити розподілені на три типи:

підзапити, що не повертають жодного або повертають декілька елементів (значень або рядків). Вони починаються з оператора IN або оператора порівняння та можуть містити ключові слова ALL, ANY (SOME);

скалярні підзапити, що повертають одне-єдине значення. Вони починаються з оператора порівняння;

підзапити, що перевіряють існування певних даних за умовами. Вони починаються з ключового слова EXISTS.

Приклад 28. *Підзапити, що повертають одне значення*. Визначте перелік товарів, ціна яких більша за середню ціну всіх товарів.

```
SELECT Товар, Ціна
FROM Товари
WHERE Ціна > (
    SELECT AVG (Ціна)
    FROM Товари
    )
```

Приклад 29. *Використання підзапитів у реченні HAVING*. Визначте товари, які постачалися за накладними більше разів, ніж кекс "Ласунка". Тут мається на увазі скільки разів підприємства-виробники постачали такий товар, а не його загальна кількість у штуках.

```
SELECT Товар, COUNT(Н.Номер_накладної) AS Кількість
FROM Накладні AS Н, ТовариНакладних AS ТН, Товари AS Т
WHERE Н.Номер_накладної=ТН.Номер_накладної AND
```

```
    TH.Код_товару=T.Код_товару
GROUP BY Товар
HAVING (COUNT(H.Номер_накладної) > (
    SELECT COUNT(H1.Номер_накладної)
    FROM Накладні H1, ТовариНакладних TH1, Товари T1
    WHERE H1.Номер_накладної=TH1.Номер_накладної AND
        TH1.Код_товару=T1.Код_товару AND
        T1.Товар= 'Кекс "Ласунка"'
    ));
```

Приклад 30. *Підзапити, що повертають декілька значень*. Визначте назви виробників, що постачали товари 20 січня 2021 року.

```
SELECT Виробник
FROM Виробники
WHERE (Код_виробника IN (
    SELECT Код_виробника
    FROM Накладні
    WHERE (Дата='20/01/2021'))
    )
```

Приклад 31. *Підзапити, що повертають декілька значень (використовують ALL, ANY або SOME)*. Визначте товари, ціна реалізації яких вища за ціну будь-яких товарів групи "Хліб"

```
SELECT Товар, Ціна
FROM Товари T
WHERE (Ціна > ALL (
    SELECT Ціна
    FROM Товари T, ГрупиТоварів ГТ
    WHERE (T.Код_групи=ГТ.Код_групи AND Група='Хліб' )
    ))
```

Приклад 32. *Підзапити, що виконують перевірку на існування*. Визначте назви товарів, що були реалізовані 2 лютого 2021 року.

```
SELECT Товар
FROM Товари
WHERE ( EXISTS(
    SELECT Код_товару
    FROM Продажі
    WHERE (Товари.Код_товару=Продажі.Код_товару AND Дата='02/02/2021')
    ))
```

Приклад 33. *Порівняння за декількома значеннями*. Отримайте назви товарів, які мають мінімальну ціну у своїй групі.

```
SELECT Товар
FROM Товари
WHERE ( Код_групи, Ціна) IN (
    SELECT Код_групи, Min(Ціна)
    FROM Товари
    GROUP BY Код_групи
    )
```

Практична частина

Попередні зауваження

Завдання лабораторної роботи виконуються з базою даних **Торгівля**, яка містить таблиці **Замовники**, **Продавці** та **Замовлення** у середовищі *Visual Studio*. Для цього слід скопіювати папку з проектом, що створений у лабораторній роботі 1.

Запити виконують у вікні редактора T-SQL. Подання додають до папки **Views** відповідної бази даних контекстною командою **Add New View**.

Якщо для виконання запитів будуть потрібні додаткові таблиці або нові поля в існуючих таблицях, їх слід створити. Дані слід підібрати так, щоб у результаті виконання запиту, який може містити кілька рядків, їх було не менше двох.

Завдання виконуються за варіантами відповідно до табл. 2.1.

Таблиця 2.1

Варіанти завдань

Варіанти	Завдання		
	1	2	3
1	2	3	4
1	1;4;7;10;13	1;3;6;10;12;15	1;3;7;10;14
2	2;5;8;10;14	2;4;7;11;13;16	2;4;8;11;15
3	3;6;9;11;13	1;5;8;10;14;17	1;5;9;12;14
4	1;4;8;11;14	2;3;9;11;12;18	2;4;6;13;15
5	2;5;7;12;13	1;4;6;10;13;17	1;3;7;12;15
6	3;6;8;12;14	2;5;7;11;14;16	2;4;8;11;14
7	1;5;9;10;13	1;4;8;10;12;15	1;5;9;10;15
8	2;6;9;11;14	2;5;9;11;13;18	2;5;6;10;14
9	3;4;8;12;13	1;3;6;10;14;17	1;4;7;11;14
10	1;5;7;12;14	2;4;7;11;13;16	2;3;8;11;15

Закінчення табл. 2.1

1	2	3	4
11	2;6;7;11;13	1;5;8;10;12;15	1;3;9;12;15
12	3;4;9;10;14	2;3;9;11;14;16	2;4;8;12;14
13	1;6;8;11;13	1;5;8;10;13;17	2;5;7;13;14
14	2;4;7;12;14	2;3;7;11;12;18	1;4;6;13;15
15	3;5;8;10;13	1;4;6;10;13;15	2;3;7;10;15
16	1;6;9;11;14	2;5;9;11;14;16	1;3;9;11;14
17	2;4;7;12;13	1;3;8;10;12;17	2;4;8;12;14
18	3;5;7;11;14	2;4;7;11;13;18	1;4;6;13;15
19	1;4;8;10;13	1;4;6;10;13;17	2;5;9;11;15
20	2;5;8;10;14	2;5;9;11;12;16	1;5;8;12;15
21	3;6;9;11;13	1;3;8;10;14;15	2;4;7;11;14
22	1;5;9;12;14	2;4;7;11;13;16	1;3;6;10;14
23	2;6;7;12;13	1;5;6;10;12;17	2;5;8;10;15
24	3;4;8;11;14	2;3;7;11;14;18	1;4;7;11;14
25	1;6;9;10;13	1;4;8;10;13;15	2;5;9;11;15
26	2;4;8;11;14	2;5;9;11;12;161	1;3;6;12;14
27	3;5;7;12;13	1;4;8;10;14;17	2;4;8;12;14
28	1;5;9;11;14	2;3;7;11;12;18	1;3;9;13;15
29	2;4;8;10;13	1;4;6;10;13;17	2;5;7;13;15
30	3;6;7;12;14	2;5;9;11;14;16	1;4;6;10;14

2.1. Побудова базових запитів

Завдання

1. Виведіть номери замовлень, суму та дату для всіх рядків з таблиці **Замовлення**.

2. Виведіть всі рядки з таблиці **Замовлення**, у яких код продавця дорівнює 4.

3. Виведіть таблицю зі стовпцями **Місто**, **Прізвище_продавця**, **Код_продавця** і **Комісійні**.

4. Виведіть рейтинг і прізвище кожного замовника з міста **Київ**.

5. Виведіть усі коди продавців з таблиці **Замовлення** без повторень.

6. Виведіть прізвища продавців з міста **Харків**, у яких комісійні перевищують 0.1.

7. Виведіть список усіх замовників з рейтингом не вище 80, крім тих, які живуть у Києві.

8. Що буде виведено в результаті виконання такого запиту?

```
SELECT * FROM Замовлення
WHERE (Сума <1000 OR
    NOT (Дата_заказа = '01.09.2018 'AND
    Код_замовника > 3));
```

9. Виведіть всі пари продавців, які живуть в одному місті. Виключіть комбінації продавців з самими собою, а також дублікати рядків, що виводяться в зворотному порядку.

10. Виведіть прізвища та міста всіх замовників з таким самим рейтингом як у Іваненка. Використовуйте поле Код_замовника для Іваненка, а не його конкретний рейтинг.

11. Визначте загальну суму замовлень на 3 вересня поточного року.

12. Підрахуйте кількість різноманітних непорожніх значень поля **Місто** в таблиці **Замовники**.

13. Виберіть прізвище замовника з найвищим рейтингом, із тих замовників, у яких прізвище починається на букву 'П'.

14. Обчисліть загальну суму комісійних, отриманих продавцями на замовлення, що зроблені замовниками з рейтингом понад 90.

2.2. Створення запитів з групуванням даних

Завдання

1. Нехай кожен продавець має комісійні в розмірі 8 %. Виведіть код замовлення, код продавця і суму комісійних для кожного замовлення.

2. Визначіть максимальний рейтинг для кожного міста. Результат повинен бути здійснений у вигляді:

Для міста **Місто** максимальний рейтинг становить: **Рейтинг**.

3. Виведіть список замовників у порядку спадання рейтингу. Прізвища замовників супроводжуйте їхнім кодом і рейтингом.

4. Виведіть загальні суми замовлень за кожен день у порядку спадання витрат на пальне.

5. Виведіть список замовників, чиї прізвища починаються на букву "В", у алфавітному порядку.

6. Визначте мінімальну суму замовлення для кожного замовника.

7. Напишіть запит для виведення всіх замовлень, що зроблені 1 або 3 вересня поточного року.

8. Виберіть всіх замовників, які обслуговуються продавцями Іваненком і Петренком.

9. Виведіть всіх замовників, чиї прізвища починаються з літери, які потрапляють в діапазон від А до К.

10. Виберіть усі замовлення, що мають нульові або невизначені значення у полі **Сума**.

11. Виберіть усіх продавців, які обслуговують замовників з рейтингом 30:

а) за допомогою предиката **EXISTS**;

б) використовуючи операцію з'єднання.

12. Виберіть з таблиці **Замовники** тих замовників, яких обслуговує продавець, який вже має хоча б одного замовника з оформленими в таблиці **Замовлення** замовленнями (замовленням).

13. Виберіть усіх замовників, рейтинг яких вищий за рейтинг будь-якого замовника з міста Київ.

14. Знайдіть всіх продавців, які в своєму місті не мають ніякого замовника (використовувати предикат **ANY** або **ALL**).

15. Виберіть усі замовлення з сумою, що перевищує будь-яку для замовників міста **Одеса**:

а) використовуючи предикат;

б) використовуючи агрегатну функцію **MAX**.

16. Створіть об'єднання з двох запитів, що відображають прізвища, міста та рейтинги всіх замовників. Замовників, які мають рейтинг 50 і більше, треба супроводжувати словами "Високий рейтинг", а інших – "Низький рейтинг".

17. Виведіть прізвища та номери кожного продавця і замовника, коли замовник має більше одного поточного замовлення. Результат подати в алфавітному порядку.

18. Сформуйте об'єднання з двох запитів. Перший запит вибирає номери всіх продавців міста Київ, а другий – номери всіх замовників з міста Київ.

2.3. Побудова запитів з підзапитами та запитів на зміну даних

Завдання

1. Використовуючи підзапит, виберіть усі замовлення замовника Іваненка.

2. Виведіть прізвища та рейтинги всіх замовників, які мають середнє значення рейтингу.

3. Знайдіть загальну суму всіх замовлень для кожного продавця, якщо ця загальна сума не перевищує суму, що має максимальну суму замовлення.

4. Використовуючи підзапит, виберіть прізвища та номери всіх замовників з максимальним для їхнього міста рейтингом.

5. Виберіть усіх продавців (із зазначенням прізвища та коду), які мають у своїх містах замовників, не обслуговуваних ними:

а) з використанням з'єднання;

б) з використанням підзапиту.

6. Помістіть такі значення в таблицю **Продавці** в зазначеному порядку: у полі **Місто** – Київ, у полі **Прізвище** – Петренко, у полі **Комісійні** – NULL, у полі Код_продавця – 11.

7. Видаліть усі замовлення замовника Сидоренка з таблиці **Замовлення**.

8. Збільште рейтинг усіх замовників міста Одеса на 10.

9. Продавець Іваненко звільнився з компанії. Передайте його замовників продавцеві Петренку.

10. Нехай є таблиця **Багато_Замовників** з такою ж структурою, як таблиця **Продавці**, і з такими ж іменами стовпців. Вставте в таблицю **Багато_Замовників** інформацію про тих продавців, які мають більше п'яти замовників.

11. Видаліть усіх замовників, які не мають поточних замовлень.

12. Збільште на 20 % комісійні тих продавців, які мають поточні замовлення на загальну суму понад 3 000 грн.

13. Визначте дати, коли загальна сума замовлень була мінімальною.

14. Визначте, які замовники не оформляли замовлення в своєму місті.

15. Визначте суму всіх замовлень за кожним містом.

2.4. Створення і використання подань

Завдання

1. Створіть подання для виведення всіх замовників з найбільшим рейтингом.

2. Створіть подання для виведення кодів продавців у кожному місті.

3. Створіть подання, яке показує середню і загальну суму замовлень для кожного продавця після його прізвища (нехай всі прізвища унікальні).

4. Створіть подання для показу продавців, які обслуговують більше одного замовника.

5. Які з наведених подань є модифікованими?

а) CREATE VIEW Просто_замовлення
AS SELECT DISTINCT Код_замовника, Код_продавця,
Код_замовлення, Дата_замовлення
FROM Замовлення;

б) CREATE VIEW Загальна_сума_замовлення
AS SELECT Прізвище_замовника, SUM (Сума)
FROM Замовлення, Замовники
WHERE Замовлення. Код_замовника =Замовники. Код_замовника
GROUP BY Прізвище_замовника;

в) CREATE VIEW Замовлення_на_третє
AS SELECT * FROM Просто_замовлення
WHERE Дата_замовлення = '03/09/2018;

д) CREATE VIEW Невизначені_міста
AS SELECT Код_продавця, Прізвище_продавця, Місто
FROM Продавці
WHERE Місто IS NULL OR
Прізвище_продавця BETWEEN 'A' AND 'Z';

6. Створіть подання для таблиці **Продавці** з ім'ям **Комісійні**. Подання має включати тільки поля **Комісійні** та **Код_продавця**. За допомогою цього подання, можна буде вводити або змінювати комісійні, але тільки ті значення, які перебувають в діапазоні від 0.10 до 0.20.

Завдання для самостійного виконання

1. Для бази даних ***Хліб*** самостійно сформулюйте такі запити та реалізуйте їх мовою SQL:

1.1. зі з'єднанням кількох таблиць, з умовою фільтрування та впорядкуванням;

1.2. з групуванням та використанням агрегатних функцій;

1.3. з підзапитами;

1.4. на зміну даних (додавання, видалення, модифікування даних);

1.5. створення подання.

2. Повторіть п. 1 для індивідуальної бази даних.

Примітка. У звіті за кожним завданням подайте текст завдання, запит на його виконання та скриншот результату.

Лабораторна робота 3
Дослідження особливостей проектування SQL-запитів засобами СУБД SQL Server

Цілі роботи

1. Набуття практичних навичок побудови збережених процедур і функцій користувача.

2. Створення і використання тригерів.

Перед виконанням роботи студент повинен знати:

основи використання *Visual Studio*;
основні об'єкти бази даних MS SQL Server;
основні команди мови T-SQL.

Після виконання лабораторної роботи студент повинен уміти:

самостійно будувати збережені процедури та функції користувача;
самостійно створювати тригери як об'єкти бази даних;
вміти тестувати збережені процедури та функції користувача, а також тригери.

Підготовча частина

Хід роботи

3.1. Збережені процедури.
3.2. Функції користувача.
3.3. Тригери.

Форма звітності

За результатами виконання роботи оформити електронний звіт засобами Word.

В електронному звіті за кожним завданням подати текст завдання, скрипт на створення відповідного об'єкта бази даних, скрипт на його тестування та скриншот результату.

Усі скриншоти з виконаними запитами для SQL Server повинні мати коментар в першому рядку з прізвищем студента.

Коментар задається рядком з двома початковими мінусами.

Критерії оцінювання

У 12-бальній системі оцінка результату захисту лабораторної роботи формується за такими правилами:

1. За кожне завдання з діапазону 3.1 – 3.3 може бути виставлено від 0 до 0,7 бала.

2. За кожне завдання з діапазону 1.1 – 1.3 із п. "Завдання для самостійного виконання" може бути виставлено від 0 до 0,5 бала, а за аналогічні завдання з індивідуальною базою даних від 0 до 0,7 бала.

3. За використання курсору під час виконання завдання додається 0,2 бала до кожного завдання.

4. За кілька варіантів виконання одного із завдань додається 0,3 бала.

5. За вибір варіанту, який з кількох варіантів виконання є оптимальним, та за обґрунтування вибору додається 0,5 бала.

6. За побудову запитів в іншій СУБД (Oracle, DB2 тощо) додається 1 бал за кожний запит.

7. За запізнення із захистом лабораторної роботи знімається 2 бали за кожний тиждень.

Отримана кількість балів з відповідей на кожне завдання лабораторної роботи підсумовується. У результаті такого підрахунку студентом може бути отримано від 0 до 12 балів.

Рекомендована література: [4; 5; 15; 19].

Основні поняття

Transact-SQL (T-SQL) – це процедурне розширення мови SQL компанією Microsoft для Microsoft SQL Server.

Стандарт мови SQL був розширений такими додатковими можливостями:

керівні оператори;

локальні та глобальні змінні;

додаткові функції для обробки рядків, дат, математики тощо;

підтримка аутентифікації *Microsoft Windows*.

Пакет – це одна або кілька команд мови T-SQL, відправлених клієнтським застосуванням у SQL Server для їх подальшого виконання у вигляді єдиного цілого.

Змінні використовують для тимчасового зберігання даних і їх подальшого застосування в рамках того самого пакету.

Змінні використовують в пакетах і скриптах:

в якості лічильника циклу;

для зберігання значення, яке необхідно перевірити інструкцією управління потоком (умовна логіка);

для зберігання значення, що повертається функцією або збереженою процедурою.

Приклад 1. *Одна змінна*. Оголосіть змінну цілого типу та надайте їй значення.

```
DECLARE @Кількість AS INT;
SET @Кількість = 10;
```

Приклад 2. *Кілька змінних*. Оголосіть дві змінні різних типів та надайте їм значення.

```
DECLARE @Товар nvarchar(25), @Ціна money;
SET @Товар = N'Хліб "Український"';
SET @Ціна = 10.50;
```

Приклад 3. *Окремий вираз*. Визначте назву товару, який має значення коду 2, і виведіть її.

```
DECLARE  @Товар nvarchar(25);
SET @Товар =
    (SELECT Товар
        FROM Товари
        WHERE Код_товару = 2);
SELECT @Товар AS Товар;
```

Приклад 4. *Використання в команді Select, одне значення*. Визначте назву товару, який має значення коду 2; відразу надайте її змінній, а потім виведіть її.

```
DECLARE  @Товар nvarchar(25);
SELECT @Товар = Товар
    FROM Товари
    WHERE Код_товару = 2;
SELECT @Товар AS Товар;
```

Приклад 5. *Використання в команді Select, кілька значень*. Визначте назву останнього товару в таблиці Товари, а потім виведіть її.

```
DECLARE  @Товар nvarchar(25);
SELECT @Товар = Товар
    FROM Товари;
SELECT @Товар AS Товар;
```

Пакет – це група з однієї або декількох інструкцій мови T-SQL, що відправляються одноразово з програми в SQL Server для виконання.

Інструкції T-SQL у пакеті закінчуються крапкою з комою, а пакети – командою GO.

Інструкції пакету виконуються у такому порядку. Спочатку сервер SQL Server компілює інструкції пакета в єдиний виконуваний модуль – план виконання. Потім інструкції в плані виконання послідовно виконуються.

Помилка компіляції (наприклад, синтаксична) припиняє компіляцію плану виконання. Тому жодна інструкція в пакеті не виконується.

Помилка виконання (наприклад, арифметичне переповнення або порушення обмеження) призводить до одного з двох результатів:

більшість помилок часу виконання зупиняють поточну інструкцію та наступні за нею в пакеті інструкції;

деякі помилки часу виконання (наприклад, порушення обмежень) зупиняють тільки поточну інструкцію. Усі інші інструкції в пакеті будуть виконані.

Це не впливає на інструкції, які виконуються до тієї, в якій зустрілася помилка виконання. Єдиний виняток – якщо пакет перебуває в транзакції. Тоді через помилки буде виконане відкочування транзакції. У цьому разі буде виконане відкочування будь-яких незафіксованих змін даних, що зроблені перед помилкою виконання.

Пакети використовують за такими правилами:

інструкції CREATE можна об'єднувати з іншими інструкціями в пакеті;

у тому самому пакеті можна спочатку змінити таблицю, а потім звернутися до нових стовпців;

якщо інструкція EXECUTE – перша в пакеті, ключове слово EXECUTE не обов'язкове. Ключове слово EXECUTE потрібне, якщо інструкція EXECUTE не є першою інструкцією в пакеті.

Інструкція IF...ELSE задає умови для виконання інструкції T-SQL. Необов'язкове ключове слово ELSE дозволяє вказати альтернативну інструкцію, яка виконується за хибного значення виразу, що йде після IF або значення NULL.

Приклад 6. *Інструкція IF...ELSE*. Для адміністратора виведіть усі дані про товари, а для покупця – тільки назву товару та ціну.

```
DECLARE @User nvarchar(10);

SET @User = 'Admin';
IF @User = 'Admin'
    SELECT * FROM Товари
ELSE
    SELECT Товар, Ціна FROM Товари;

SET @User = 'Покупець';
IF @User = 'Admin'
    SELECT * FROM Товари
ELSE
    SELECT Товар, Ціна FROM Товари;
```

Блок – це послідовність інструкцій мови T-SQL, що дозволяє виконувати групу інструкцій як одну інструкцію. Блок починається з BEGIN і закінчується END.

Приклад 7. *Блок*. Залежно від типу користувача виведіть або всі дані про товари та виробників, або тільки назву та ціну товару та назву виробника.

```
DECLARE @User nvarchar(10);
SET @User = 'Покупець';
IF @User = 'Admin'
  BEGIN
    SELECT * FROM Товари;
    SELECT * FROM Виробники
  END
ELSE
  BEGIN
    SELECT Товар, Ціна FROM Товари;
    SELECT Виробник FROM Виробники
  END;
```

Інструкція WHILE задає умову повторного виконання SQL-інструкції або блоку інструкцій. Ці інструкції викликаються в циклі, поки зазначена умова істинна.

Вона має такий формат:

```
WHILE Boolean_expression
     { sql_statement | statement_block | BREAK | CONTINUE }
```

де BREAK – вихід з найближчого циклу WHILE. Викликаються інструкції, наступні за ключовим словом END, що позначає кінець циклу;

CONTINUE – новий крок циклу, не враховуючи всі інструкції, що йдуть за CONTINUE.

Приклад 8. *Інструкція WHILE*. Заповніть датами таблицю **Дати**.

```
DECLARE @ПочатковаДата DATE;
DECLARE @КінцеваДата DATE;

SET @ПочатковаДата = '01/01/2010';
SET @КінцеваДата = '12/31/2018';

DECLARE @ПоточнаДата DATE;
SET @ПоточнаДата = @ПочатковаДата;

WHILE @ПоточнаДата <= @КінцеваДата
    BEGIN
        INSERT INTO Дати VALUES (@ПоточнаДата);
        SET @ПоточнаДата = DateAdd(d, 1, @ПоточнаДата)
    END;

select count(*) AS 'Кількість дат' from Дати;
```

Команди DML (SELECT, UPDATE, DELETE) працюють відразу з групами рядків. Для роботи з кожним відібраним рядком використовують курсори.

Курсор (current set of records) – це тимчасовий набір рядків, які можна перебирати послідовно з першого до останнього.

Для роботи з курсорами використовують такі команди.

Оголошення курсора:

```
DECLARE ім'я_курсора CURSOR FOR SELECT текст_запиту
```

Відкриття курсора:

```
OPEN ім'я_курсора
```

Читання наступного рядка з курсора:

```
FETCH ім'я_курсора INTO список_змінних
```

Примітка: глобальна змінна @@FETCH_STATUS <> 0, якщо рядків в курсорі більше немає. Якщо набір рядків ще не вичерпаний, то @@FETCH_STATUS = 0, і оператор FETCH перепише значення полів з поточного рядка в змінні.

Закриття курсора:

```
CLOSE ім'я_курсора
```

Видалення курсора з пам'яті:

```
DEALLOCATE ім'я_курсора
```

Приклад 9. *Використання курсора*. На основі даних з таблиць **Продажі** та **Товари** заповніть таблицю **таблНакопичення**, яка містить такі стовпці: Дата, Виторг та Накопичення. Поле Дата вибирається з таблиці **Продажі**; поле Виторг обчислюється як добуток кількості проданого товару на його ціну; поле **Накопичення** визначається як загальна вартість отриманих грошей на певну дату з урахуванням коштів, що були отримані до цього дня.

```
DECLARE @Виторг money, @Накопичення money;
DECLARE @Дата DATE;
DECLARE C CURSOR FOR
    SELECT Дата, SUM(Товари.Ціна*Продажі.Кількість) AS Виторг
    FROM Продажи JOIN Товари
        ON Продажі.Код_товару=Товари.Код_товару
    GROUP BY Дата;
OPEN C;
FETCH FROM C INTO @Дата, @Виторг;
SET @Накопичення = 0;
WHILE @@FETCH_STATUS = 0
    BEGIN
        SET @Накопичення = @Накопичення + @Виторг;
        INSERT INTO таблНакопичення VALUES(@Дата, @Виторг, @Накопичення);
        FETCH FROM C INTO @Дата, @Виторг;
    END;
CLOSE C;
DEALLOCATE C;
SELECT * FROM таблНакопичення;
```

Збережена процедура в SQL Server – це група з однієї або декількох інструкцій T-SQL. Вона зберігається на сервері, щоб поліпшити продуктивність і сталість виконання повторюваних завдань.

Збережена процедура попередньо компілюється перед тим, як вона буде збережена у вигляді об'єкта бази даних. Під час кожного виклику використовується вже відкомпільована процедура. Вона обробляє вхідні

параметри та повертає програмі, що її викликає, значення у вигляді вихідних параметрів.

Збережена процедура має такі переваги:

зниження мережевого трафіку між клієнтами та сервером;

підвищена безпека (дозволи на використання на рівні процедури, приховування операторів, захист від ін'єкцій та шифрування);

повторне використання коду;

полегшене обслуговування (в застосуванні);

підвищена продуктивність.

Збережені процедури найчастіше створюють за допомогою запиту.

Приклад 10. *Створення збереженої процедури*. Створіть збережену процедуру для знаходження вартості проданих товарів заданого виробника.

```
CREATE PROCEDURE Варість_виробника
     @Код_виробника int,
     @Загальна_вартість money OUTPUT
AS
     SELECT @Загальна_вартість = SUM(Ціна * Кількість)
       FROM Продажі
         JOIN Товари ON Продажі.Код_товару =Товари.Код_товару
       WHERE Продажі.Код_виробника = @Код_виробника
```

Збережену процедуру викликають за допомогою оператора EXEC, підставляючи замість формальних параметрів фактичні.

Приклад 11. *Виклик збереженої процедури*. Обчисліть загальну вартість проданих товарів виробника з кодом 2.

```
DECLARE @Загальна_вартість AS money;
EXEC Вартість_виробника 2, @Загальна_вартість OUTPUT;
SELECT  @Загальна_вартість;
```

На відміну від процедури функція користувача може мати тільки один результат, який з виконанням функції відразу може використовуватися у виразі. Вона має такі обмеження:

не може змінювати базу даних (містить тільки оператори SELECT);

не має вихідних параметрів.

Функцію користувача створюють за таким форматом:

```
CREATE FUNCTION <Ім'я>(<Параметри>)
RETURNS Тип_даних
AS
```

```
    BEGIN
        Блок оголошень
        Код процедури
        RETURN <Значення>
    END;
```

Приклад 12. *Створення функції користувача*. Замість збереженої процедури для знаходження вартості проданих товарів заданого виробника (див. приклад 10) створіть функцію користувача з тою самою функціональністю.

```
CREATE PROCEDURE fВарість_виробника(
     @Код_виробника int
)
RETURNS money
AS
BEGIN
     DECLARE @Загальна_вартість AS money;
     SELECT @Загальна_вартість = SUM(Ціна * Кількість)
         FROM Продажі
             JOIN Товари ON Продажі.Код_товару =Товари.Код_товару
         WHERE Продажі.Код_виробника = @Код_виробника;
     RETURN @Загальна_вартість;
END
```

Збережену процедуру викликають як операнд у виразі.

Приклад 13. *Виклик функції користувача*. Порівняйте загальну вартість проданих товарів виробників з кодом 1 і 2.

```
IF dbo.fВарість_виробника(1) > dbo.fВарість_виробника (2)
PRINT N'Вартість проданих товарів першого виробника більша, ніж другого'
ELSE
PRINT N'Вартість проданих товарів другого виробника більша, ніж першого';
```

Тригер – це спеціальний вид збережених процедур, який спрацьовує та запускає свій код у момент, коли відбувається певна подія.

SQL Server підтримує два *типи подій*:

операції зміни даних (DML-тригери);

зміна структури бази даних (DDL-тригери).

Тригери можуть спрацьовувати або після (AFTER) завершення відповідної події, або замість неї (INSTEAD OF).

Тригер виконується автоматично під час виконання операцій оновлення, додавання або видалення даних.

Кожен тригер прив'язується до конкретної таблиці.

Усі зміни даних розглядаються як одна транзакція. У разі виявлення помилки або порушення цілісності даних відбувається відкочування цієї транзакції. Тим самим внесення змін забороняється. Скасовуються також усі зміни, вже зроблені тригером.

Тригери DML використовують логічні таблиці deleted та inserted. За своєю структурою вони подібні до таблиці, на якій визначено тригер, тобто до якої застосовується дія користувача. У таблицях deleted та inserted містяться старі або нові значення рядків, які можуть бути змінені діями користувача.

Тригер створюють за таким форматом:

```
CREATE TRIGGER <Ім'я>
ON <Ім'я_таблиці>
AFTER INSERT
AS
    BEGIN
        Блок оголошень
        Код процедури
        RETURN <Значення>
    END;
```

Приклад 14. *Створення тригера*. Нехай в таблицю **Товари** додано поле Продано, яке містить кількість проданого товару з кодом, що визначає певний рядок. Створіть тригер, який після додавання рядка в таблицю **Продажі** збільшує значення поля **Продано** відповідного товару у таблиці **Товари**.

```
CREATE TRIGGER trgПродажіПродано
ON Продажі
AFTER INSERT
AS
BEGIN
    DECLARE @Код_товару AS int;
  DECLARE @Кількість AS smallint;

    SELECT @Код_товару =(SELECT Код_товару
        FROM inserted);
    SELECT @Кількість =(SELECT Кількість
```

```
        FROM inserted);

    UPDATE Товари
        SET Продано = Продано + @Кількість
        WHERE Код_товару= @Код_товару;
END
```

Практична частина

Попередні зауваження

Завдання лабораторної роботи виконуються з базою даних **Торгівля**, яка містить таблиці **Замовники**, **Продавці** та **Замовлення** у середовищі *Visual Studio*. Для цього слід скопіювати папку з проектом, що створений у лабораторній роботі 2.

Збережені процедури та функції додають у вікні **Server Explorer** до папок **Stored Procedures** і **Functions** відповідної бази даних контекстовими командами **Add New Stored Procedure** і **Add New Scalar-valued Functions**. Тригери додають до таблиць контекстовою командою **Add New Trigger**. Але простіше їх створити за допомогою відповідного запиту. Саме цей спосіб створення об'єктів бази даних слід вибрати для розв'язання завдань.

Перед тим як створити збережену процедуру чи функцію користувача, зазвичай попередньо будують скрипт без процедури, щоб відпрацювати алгоритм. Потім готові конструкції перетворюють на збережену процедуру чи функцію користувача.

Наприклад, для обчислення загальної суми замовлень за вказаний день спочатку записують такий скрипт:

```
Declare @Date date,
          @Sum money;
SET LANGUAGE RUSSIAN;
Set @Date='11.10.2012';

SELECT @Sum=Sum(Сумма) From Замовлення
     Where ДатаЗамовлення=@Date;

Select @Sum;
```

Щоб перевірити його правильність, скрипт запускають на виконання. Якщо результат співпадає з попередньо обчисленим, текст перетворюють на збережену процедуру чи функцію користувача. Збережена процедура матиме такий вигляд:

```
CREATE PROCEDURE [dbo].[SumOrderOnDate]
      @Date date,
      @Sum money OUTPUT
AS
      SELECT @Sum=Sum(Сумма)
      From Заказ
      Where ДатаЗамовлення=@Date;
```

Перевіряють описану збережену за допомогою такого скрипта:

```
Declare @S1 money;
SET LANGUAGE RUSSIAN;

EXEC SumOrderOnDate '11.10.2012', @S1 output;

SELECT @S1;
```

Примітки:

1) для викликання збереженої процедури використовують оператор EXEC (скорочена форма), або EXECUTE (повна форма);

2) під час викликання збереженої процедури записують службове слово output після вихідного параметра;

3) для збереження значення вихідного параметра описують змінну (у цьому випадку @S1).

Функцію користувача можна викликати як частину виразу у запиті. Наприклад, для знаходження мінімального рейтингу замовника в указаному місті можна використати функцію користувача, яка описується таким скриптом:

```
CREATE FUNCTION [dbo].[MinRatingCity]
(
      @City nchar(20)
)
RETURNS smallint
AS
BEGIN
      DECLARE @Rating smallint;

      SELECT @Rating=MIN(Рейтинг)
            FROM Замовники
            WHERE Місто=@City;


      RETURN @Rating
END
```

Перевірку описаної функції користувача можна здійснити таким скриптом:

```
SELECT dbo.MinRatingCity(N'Київ');
```

Робота тригера перевіряється під час виконання завдання, для обробки якого він призначений.

Якщо для виконання завдань будуть потрібні додаткові таблиці або нові поля в існуючих таблицях, їх слід створити. Дані підібрати так, щоб у результаті виконання запиту, який може містити кілька рядків, їх було не менше двох.

Завдання виконуються за варіантами відповідно до табл. 3.1.

Таблиця 3.1

Варіанти завдань

Варіанти	Завдання		
	1	2	3
1	2	3	4
1	1;4;7;10;13	1;3;6;10;12;15	1;3;7;10;14
2	2;5;8;10;14	2;4;7;11;13;16	2;4;8;11;15
3	3;6;9;11;13	1;5;8;10;14;17	1;5;9;12;14
4	1;4;8;11;14	2;3;9;11;12;18	2;4;6;13;15
5	2;5;7;12;13	1;4;6;10;13;17	1;3;7;12;15
6	3;6;8;12;14	2;5;7;11;14;16	2;4;8;11;14
7	1;5;9;10;13	1;4;8;10;12;15	1;5;9;10;15
8	2;6;9;11;14	2;5;9;11;13;18	2;5;6;10;14
9	3;4;8;12;13	1;3;6;10;14;17	1;4;7;11;14
10	1;5;7;12;14	2;4;7;11;13;16	2;3;8;11;15
11	2;6;7;11;13	1;5;8;10;12;15	1;3;9;12;15
12	3;4;9;10;14	2;3;9;11;14;16	2;4;8;12;14
13	1;6;8;11;13	1;5;8;10;13;17	2;5;7;13;14
14	2;4;7;12;14	2;3;7;11;12;18	1;4;6;13;15
15	3;5;8;10;13	1;4;6;10;13;15	2;3;7;10;15
16	1;6;9;11;14	2;5;9;11;14;16	1;3;9;11;14
17	2;4;7;12;13	1;3;8;10;12;17	2;4;8;12;14
18	3;5;7;11;14	2;4;7;11;13;18	1;4;6;13;15
19	1;4;8;10;13	1;4;6;10;13;17	2;5;9;11;15
20	2;5;8;10;14	2;5;9;11;12;16	1;5;8;12;15

Закінчення табл. 3.1

1	2	3	4
21	3;6;9;11;13	1;3;8;10;14;15	2;4;7;11;14
22	1;5;9;12;14	2;4;7;11;13;16	1;3;6;10;14
23	2;6;7;12;13	1;5;6;10;12;17	2;5;8;10;15
24	3;4;8;11;14	2;3;7;11;14;18	1;4;7;11;14
25	1;6;9;10;13	1;4;8;10;13;15	2;5;9;11;15
26	2;4;8;11;14	2;5;9;11;12;16	1;3;6;12;14
27	3;5;7;12;13	1;4;8;10;14;17	2;4;8;12;14
28	1;5;9;11;14	2;3;7;11;12;18	1;3;9;13;15
29	2;4;8;10;13	1;4;6;10;13;17	2;5;7;13;15
30	3;6;7;12;14	2;5;9;11;14;16	1;4;6;10;14

3.1. Збережені процедури

Завдання

Побудуйте та перевірте збережені процедури для виконання таких завдань.

1. Виведіть номери замовлень, суму та дату для всіх рядків з таблиці **Замовлення** вказаного замовника.

2. Виведіть всі рядки з таблиці **Замовлення**, у яких брав участь указаний продавець.

3. Виведіть таблицю зі стовпцями **Прізвище_продавця** і **Комісійні**, що працюють у вказаному місті.

4. Виведіть рейтинг і прізвище кожного замовника із вказаного міста.

5. Виведіть всі коди продавців з таблиці **Замовлення** без повторень за вказаний проміжок часу.

6. Виведіть прізвища продавців із вказаного міста, в яких комісійні перевищують 0.1.

7. Виведіть список усіх замовників з рейтингом не вище вказаного, крім тих, які живуть в зазначеному місті.

8. Виведіть всі дані про замовлення з сумою, що не перевищує вказану, за зазначений період.

9. Виведіть всі пари продавців, які живуть у зазначеному місті за вказаний період; виключіть комбінації продавців з самими собою, а також дублікати рядків, що виводяться в зворотному порядку.

10. Виведіть прізвища та міста всіх замовників з таким самим рейтингом як у вказаного, які зробили замовлення у зазначений період.

11. Визначте загальну суму замовлень на вказану дату в зазначеному місті.

12. Підрахуйте кількість замовлень, що зроблені у зазначеному місті на вказаний діапазон сум.

13. Виведіть прізвища, що починаються із заданої букви, замовників які зробили замовлення у зазначеному місті.

14. Обчисліть загальну суму комісійних, отриманих продавцями на замовлення у вказаний період, що зроблені замовниками, рейтинг яких перевищує зазначену величину.

3.2. Функції користувача

Завдання

Побудуйте та перевірте функції користувача для виконання таких завдань.

1. Визначте загальну суму комісійних для вказаного продавця.

2. Визначте максимальний рейтинг замовників для вказаного міста.

3. Визначте кількість замовників для вказаного міста.

4. Визначте максимальні комісійні для вказаного міста.

5. Визначте найдовше прізвище замовника, яке починається на вказану букву.

6. Визначте мінімальну суму замовлення для вказаного замовника.

7. Визначте прізвище продавця, який оформив замовлення у вказаний день на найбільшу суму.

8. Визначте кількість замовників, які обслуговувалися вказаним продавцем.

9. Виведіть кількість замовників, чиї прізвища починаються з літер, які потрапляють в діапазон вказаних літер від Літера1 до Літера2.

10. Визначте кількість замовлень, що мають нульові або невизначені значення у полі **Сума**.

11. Визначте кількість продавців, які обслуговують замовників із указаним рейтингом.

12. Визначте кількість замовників із вказаним рейтингом, яких обслуговує продавець із зазначеного міста.

13. Визначте кількість замовників, рейтинг яких вищий за рейтинг будь-якого замовника із вказаного міста.

14. Знайдіть кількість продавців, які в своєму місті не мають жодного замовника.

15. Визначте кількість замовлень з сумою, що перевищує будь-яку суму для замовників із вказаного міста:

16. Визначте місце зазначеного продавця у списку, де продавці розташовані за спаданням розміру комісійних.

17. Визначте прізвище продавця, який оформив замовлення на найбільшу суму для зазначеного замовника.

18. Визначте місце у рейтингу зазначеного замовника для вказаного міста.

3.3. Тригери

Завдання

Створіть та перевірте тригери для виконання таких завдань.

1. Після додавання нового замовлення рейтинг замовника збільшується на один відсоток. Він не може бути більше 100.

2. Після вилучення замовлення рейтинг замовника зменшується на три відсотка. Він не може бути менше нуля.

3. Після додавання нового замовлення комісійні продавця зменшуються на два відсотка. Вони не можуть бути менше нуля.

4. Після додавання нового замовника на одиницю збільшується кількість замовників у його місті (таблиця **Міста**).

5. Після додавання нового продавця на одиницю збільшується кількість продавців у його місті (таблиця **Міста**).

6. Після вилучення замовника на одиницю зменшується кількість замовників у його місті (таблиця **Міста**).

7. Після вилучення продавця на одиницю зменшується кількість продавців у його місті (таблиця **Міста**).

8. Після додавання нового замовлення на одиницю збільшується кількість замовників, яких обслуговує даний продавець (якщо раніше він не обслуговував доданого замовлення).

9. Після додавання нового замовлення збільшується загальна сума вартості замовлень, які оформляв продавець.

10. Після вилучення замовлення у подвійному розмірі зменшується загальна сума вартості замовлень, які оформляв продавець. Вона не може бути менше нуля.

11. Після вилучення замовлення зменшується загальна сума вартості замовлень, які здійснював замовник. Вона не може бути менше нуля.

12. Після видалення продавця з таблиці **Продавці**, передати всі його замовлення продавцю з того самого міста, у якого найменше замовників.

13. Після зміни суми замовлення у таблицю **Корегування** заносяться всі дані про замовлення із зазначенням початкової та кінцевої сум.

14. Після зміні міста замовника на одиницю зменшується кількість замовників у попередньому місті, а також на одиницю збільшується кількість замовників в новому(таблиця **Міста**).

15. Після додавання нового замовлення збільшується загальна сума вартостей замовлень за поточний день, які оформляв продавець (таблиця **ПродажіПродавців**).

Завдання для самостійного виконання

1. Для бази даних **Хліб** самостійно сформулюйте такі завдання та реалізуйте їх мовою SQL.

1.1. з використанням збережених процедур.

1.2. з використанням функцій користувача.

1.3. з використанням тригерів.

2. Повторіть п. 1 для індивідуальної бази даних.

3. Виконайте попередні завдання для самостійного виконання у середовищі *Management Studio* і порівняйте з аналогічними засобами *Visual Studio*.

Примітки:

1) завдання 1.1 і 1.2 сформулюйте так, щоб використовувалися збережені процедури та функції:

а) без параметрів;

б) з вхідними та вихідними параметрами;

2) у звіті з лабораторної роботи за кожним завданням подайте текст завдання, скрипт на його створення, запит на виконання та скриншот результату.

Лабораторна робота 4
Нормалізація відношень у базах даних

Цілі роботи

1. Вивчити поняття функціональної і багатозначної залежностей.
2. Вивчити технологію нормалізації реляційної бази даних.
3. Засвоїти операції реляційної алгебри.

Перед виконанням роботи студент повинен знати:

основи проектування реляційних баз даних;
основи теорії множин.

Після виконання лабораторної роботи студент повинен уміти:

самостійно складати формалізований опис предметної області;
самостійно встановлювати зв'язки між таблицями бази даних;
самостійно створити схему бази даних.

Підготовча частина

Хід роботи

4.1. Базові нормальні форми.
4.2. Операції реляційної алгебри.
4.3. Старші нормальні форми.
4.4. Денормалізація.

Форма звітності

За результатами виконання роботи оформити електронний звіт засобами Word. В електронному звіті за кожним завданням подати текст завдання з коментарями за формою, що подана в наведених нижче прикладах.

Критерії оцінювання

У 12-бальній системі оцінка результату захисту лабораторної роботи формується за такими правилами:

1. За завдання 4.1 може бути виставлено від 0 до 4 балів.
2. За завдання 4.2 може бути виставлено від 0 до 2 балів.
3. За завдання 4.3 може бути виставлено від 0 до 4 балів.
4. За завдання 4.4 може бути виставлено від 0 до 2 балів.
5. За запізнення із захистом лабораторної роботи знімається 2 бали за кожний тиждень.

Отримана кількість балів з відповідей на кожне завдання лабораторної роботи підсумовується. У результаті такого підрахунку студент може отримати від 0 до 12 балів.

Рекомендована література: [10; 11].

Основні поняття

Предметна область – частина реального світу, що моделюється із застосуванням баз даних.

Сутність – будь-який помітний об'єкт (об'єкт, що можна відрізнити від іншого), інформацію про який необхідно зберігати в базі даних.

Атрибут – пойменована характеристика сутності.

Домен – набір припустимих значень для одного або декількох атрибутів.

Відношення R ступеня n (n-арне відношення R) – підмножина декартового добутку множин D_1, D_2,…, D_n (n ≥ 1), де Di – домен (i = 1,2,…,n). Відношення має просту графічну інтерпретацію, воно може буде подане у вигляді таблиці.

Ступінь (розмірність) відношення – число (кількість) стовпців у відношенні.

Нехай X і Y – підмножини атрибутів з R. Y функціонально залежить від X у тому та тільки в тому випадку, якщо кожному значенню X відповідає в точності одне значення Y: R.X → R.Y (X і Y можуть бути складовими).

X – детермінант функціональної залежності, тобто атрибут, від якого залежить інший атрибут цієї ж таблиці.

Функціональна залежність R.X → R.Y є *повною*, якщо жоден атрибут не може бути опущений з її детермінанта без порушення цієї залежності.

Функціональна залежність R.X → R.Y буде *транзитивною*, якщо існують: атрибут Z, якщо є функціональні залежності R.X → R.Z і R.Z → R.Y і відсутня функціональна залежність R.Z → R.X.

Потенційний ключ – атрибут або множина атрибутів, що єдиним чином ідентифікує кортеж даного відношення.

Первинний ключ – потенційний ключ, що обраний для унікальної ідентифікації кортежів усередині відношення. *Неключовим атрибутом* називають будь-який атрибут відношення, що не входить до складу первинного ключа.

Багатозначна залежність – залежність між трьома атрибутами, коли атрибутом A визначається кінцева множина значень атрибута B і кінцева множина значень атрибута C, але атрибути B і C один від одного не залежать. *Детермінант функціональної залежності* – атрибут, від якого залежить інший атрибут цієї ж таблиці.

З'єднання – операція реляційної алгебри, в якій два відношення об'єднуються за відповідністю рядків на основі значень стовпців двох таблиць. Відношенням відповідності зазвичай є комбінація "первинний ключ – зовнішній ключ".

Нормалізація – процес розміщення атрибутів у таблицях, що виключає проблеми, характерні для хибно організованої бази даних. *Нормальна форма* – критерії проекту, яким повинно відповідати відношення.

Денормалізація – модифікація реляційної БД, за якої порядок її нормалізації зменшується. Денормалізація: дублювання неключових атрибутів для зменшення кількості з'єднань, а також створення таблиць даних, які містяться в інших сутностях для об'єднання таблиць із зв'язками 1:1.

Практична частина

4.1. Базові нормальні форми

Попередні зауваження

Кожний студент вибирає опис предметної області за номером згідно зі своїм номером у журналі групи.

Якщо під час нормалізації виявляється, що база даних знаходиться у більш високій нормальній формі, потрібно модифікувати постановку задачі так, щоб студент міг показати вміння поступово виконувати нормалізацію, не пропускаючи жодного кроку.

4.1.1. Постановка задачі

Завдання

Сформулюйте постановку задачі в термінах предметної області. Укажіть, ким за посадою, коли і як часто будуть використовуватися дані, що зберігаються в базі.

Приклад виконання

І. Постановка задачі (Реалізація хлібобулочних виробів)

Існує ряд підприємств, що випускають різноманітну хлібобулочну продукцію. Кожен вид продукції випускається відповідно до державних стандартів (ДСТУ). Щодня згідно з укладеними договорами на поставку продукції вироблена продукція відпускається фірмам-реалізаторам, які безпосередньо продають хлібобулочні вироби населенню. За отриману продукцію фірма-реалізатор перераховує відповідні кошти на банківський розрахунковий рахунок фірми-виробника.

4.1.2. Словник даних

Завдання

На основі аналізу постановки задачі побудуйте словник даних, указавши для кожного елемента даних, що буде зберігатися в базі даних, його ідентифікатор, тип даних і призначення. Словник даних подайте у вигляді табл. 4.1.

Приклад виконання

У базі даних **"Постачання хлібобулочних виробів"** будуть зберігатися елементи даних, що подані у табл. 4.1.

Таблиця 4.1

Словник даних

№ п/п	Найменування елемента	Ідентифікатор	Тип і довжина	Призначення елемента
1	Виробник	Postavshik	VARCHAR (30)	Найменування фірми виробника ХБВ
2	Реалізатор	Realizator	VARCHAR (30)	Найменування фірми реалізатора ХБВ
3	№ договора	No_Dogovora	VARCHAR (6)	Номер договору на поставку ХБВ
4	Банк виробника	Bank_Postav	VARCHAR (30)	Найменування банку виробника
5	Р/Р виробника	RS_Postav	NUMBER (8,0)	Номер розрахункового рахунку виробника в банку
6	МФО банку виробника	MFO_Banka_Postav	NUMBER (6,0)	МФО банку виробника
7	Товар	Tovar	VARCHAR (30)	Найменування ХБВ
8	ДСТУ на товар	DSTU_Tovar	VARCHAR (10)	Номер державного стандарту на виробництво ХБВ відповідного типу
9	Дата поставки	Data_Postavki	DATE	Дата поставки ХБВ фірмі-реалізатору
10	Кількість	Kolvo	INTEGER	Обсяг (в шт.) поставленої хлібобулочної продукції відповідного виду

4.1.3. Обмеження даних

Завдання

Визначте обмеження для можливих значень атрибутів.

Примітка. У рамках визначеного переліку атрибутів необхідно виявити обмеження, яким повинні задовольняти значення цих атрибутів. Це можуть бути обмеження на обов'язковість чи необов'язковість значення, на умови, яким повинно відповідати значення, на співвідношення між значеннями різних атрибутів тощо.

Приклад виконання

На атрибути, що зберігаються в базі даних **"Постачання хлібобулочних виробів"**, накладаються обмеження, що подані у табл. 4.2.

Таблиця 4.2

Обмеження даних

№ п/п	Найменування атрибута	Обмеження
1	Виробник	Поле повинно бути обов'язковим (NOT NULL). Значення має містити не більше 25 символів
2	Реалізатор	Поле повинно бути обов'язковим (NOT NULL). Значення має містити не більше 25 символів
3	№ Договору	Поле повинно бути обов'язковим (NOT NULL). Значення має містити 4 цифри
4	Банк виробника	Поле повинно бути обов'язковим (NOT NULL). Значення має містити не більше 25 символів
5	Р/Р виробника	Поле повинно бути обов'язковим (NOT NULL). Значення має містити 8 цифр
6	МФО банку виробника	Поле повинно бути обов'язковим (NOT NULL). Значення має містити 6 цифр
7	Товар	Поле повинно бути обов'язковим (NOT NULL). Значення має містити не більше 20 символів
8	ДСТУ на товар	Поле повинно бути обов'язковим (NOT NULL). Значення має містити 6 символів. Приклад: 123-34
9	Дата поставки	Поле повинно бути обов'язковим (NOT NULL). Значення введеної дати не може бути більше поточної. Рік дати повинен дорівнювати поточному року
10	Кількість	Поле повинно бути обов'язковим (NOT NULL). Значення позитивне

Примітка: усі обмеження на значення атрибутів і їх унікальність слід пояснити.

4.1.4. Функціональні залежності між атрибутами

Завдання

Визначте залежності між атрибутами або групами атрибутів, при цьому передбачте можливі транзитивні залежності на основі аналізу предметної області.

Приклад виконання

Між атрибутами, що зберігаються в базі даних **"Постачання хлібобулочних виробів"**, існують функціональні залежності, що подані у табл. 4.3.

Примітка: це не єдині функціональні залежності предметної області. Наприклад, можна сказати, що саме номер договору **(№ договору**) визначає значення атрибутів **Виробник** і **Реалізатор**, або **Виробник** залежить від розрахункового рахунку (**Р/Р виробника**) та банку виробника (**Банк виробника**) тощо.

Таблиця 4.3

Функціональні залежності між атрибутами

№ п/п	Функціональна залежність
1	Номер договору залежить від значень атрибутів: Виробник і Реалізатор
2	Номер ДСТУ на товар залежить від виду хлібобулочного виробу товару (Товар)
3	Розрахунковий рахунок (Р/Р виробника), Банк виробника і МФО банку виробника залежать від виробника (Виробник)
4	МФО банку виробника залежить від банку

4.1.5. Вихідне відношення

Завдання

Складіть таблицю з даними, які містять стовпці всіх атрибутів, визначених у словнику даних (не менше 20 рядків).

Приклад виконання

Дане відношення (табл. 4.4) відповідає початковому опису предметної області та відображає такі факти:

існує ряд підприємств, що випускають хлібобулочну продукцію (**Виробник**);

є множина фірм, що реалізують хлібобулочну продукцію (**Реалізатор**);

у виробників укладені договори на поставку продукції з реалізаторами, які перераховують гроші за отриманий товар на розрахунковий рахунок виробника, що знаходиться у відповідному банку;

щодня виробники постачають реалізаторам різну хлібобулочну продукцію (**Товар**), що випускається відповідно до вимог Державного стандарту (**ДСТУ на товар**) у певній кількості.

Таблиця 4.4

Постачання хлібобулочних виробів

Виробник	Реалізатор	№ договору	Р/Р виробника	Банк виробника	МФО банка виробника	Товар	ДСТУ на товар	Дата поставки	Кіль-кість
АТ "Кулиничі"	ТФ "Барс"	12-091	26000001	Синтез	851300	Дарницький	4583	10.10.21	200
АТ "Кулиничі"	ТФ "Булка"	045-11	26000001	Синтез	851300	Нарізний	4583	10.10.21	250
ЗАТ "Хліб"	ТФ "Булка"	23-011	26010234	Ексімбанк	851301	Нарізний	4583	10.10.21	350
ЗАТ "Хліб"	ТФ "Булка"	23-011	26010234	Ексімбанк	851301	Сімейка	4585	10.10.21	200
ЗАТ "Хліб"	ТФ "Здоба"	114-01	26010234	Ексімбанк	851300	Сімейка	4585	10.10.21	350
ТОВ "Салтів-ський"	ТФ" Здоба"	14-235	26000234	Синтез	851300	Білий	4583	10.10.21	245
АТ "Кулиничі"	ТФ "Барс"	12-091	26000001	Синтез	851300	Нарізний	4583	11.10.21	500
ЗАТ "Хліб"	ТФ "Булка"	23-011	26010234	Ексімбанк	851301	Український	4583	11.10.21	150
ТОВ "Салтів-ський"	ТФ" Здоба"	14-235	26000234	Синтез	851300	Житній	4583	11.10.21	200
ТОВ "Салтів-ський"	ТФ "Булка"	045-13	26000234	Синтез	851300	Завиток	4585	12.10.21	300
ЗАТ "Хліб"	ТФ "Здоба"	114-01	26010234	Ексімбанк	851301	Сімейка	4585	12.10.21	120
ТОВ "Салтів-ський"	ТФ "Здоба "	14-235	26000234	Синтез	851300	Завиток	4585	12.10.21	310

4.1.6. Перша нормальна форма

Завдання

Доведіть, що табл. 4.4 знаходиться у першій нормальній формі (1НФ). В іншому разі змініть дані так, щоб сформована таблиця перебувала в 1НФ.

Приклад виконання

Оскільки відношення знаходиться в 1НФ, воно повинно мати первинний ключ, значення якого однозначно визначає кожен кортеж.

Для даного відношення первинний ключ є складеним і включає такі атрибути:

Первинний ключ (ПК) – ПК [Виробник, Реалізатор, Товар, Дата поставки].

Аналізуючи наведене відношення, визначте залежності між атрибутами або групами атрибутів у ньому.

Від первинного ключа залежить тільки значення атрибута **Кількість**, яке характеризує обсяг закупленого реалізатором (**Реалізатор**) у виробника (**Виробник**) товару (**Товар)** на конкретну дату поставки (**Дата поставки)**. Решта атрибутів не залежить від первинного ключа. Однак є ще такі функціональні залежності, що визначені в постановці завдання (табл. 4.5).

Таблиця 4.5

Відношення в 1НФ

R	X	ФЗ	Y
R1	{Виробник, Товар, Реалізатор, Дата поставки}	->	{Кількість}
R2	{Виробник, Реалізатор}	->	{№ Договору }
R3	{Товар}	->	{ДСТУ на товар}
R4	{Виробник}	->	{Р/Р виробника, Банк виробника, МФО банку виробника}

4.1.7. Друга нормальна форма

Завдання

Визначте залежності неключових атрибутів від первинного ключа і декомпозуйте таблицю так, щоб перевести вихідне відношення в другу нормальну форму (2НФ).

Приклад виконання

Для перетворення цього відношення з 1НФ в 2НФ, необхідно декомпозувати відношенні так, щоб в усіх отриманих відношеннях кожен його атрибут, що не є основним атрибутом, функціонально повно залежав від первинного ключа цього відношення. Для цього усі детермінанти виявлених функціональних залежностей повинні бути первинними ключами.

Таким чином, вихідне відношення в 1НФ **"Постачання хлібобулочних виробів"**, перетворюється в 4 відношення у 2НФ (табл. 4.6):

Таблиця 4.6

Постачання хлібобулочних виробів

R	ПК	Залежні атрибути
"Закупівлі" (Виробник, Товар, Реалізатор, Дата поставки, Кількість)	ПК[Виробник, Товар, Реалізатор, Дата поставки]	Кількість
"Договори на постачання" (Виробник, Реалізатор, № договору)	ПК[Виробник, Реалізатор]	№ договору
"Стандарти на товари" (Товар, ДСТУ на товар)	ПК[Товар]	ДСТУ на товар
"Банківські реквізити виробників" (Виробник, Р/Р виробника в банку, Банк виробника, МФО банку виробника)	ПК[Виробник]	Р/Р виробника в банку, Банк виробника, МФО банку виробника

4.1.8. Третя нормальна форма

Завдання

На основі аналізу наявних транзитивних залежностей переведіть відношення в третю нормальну форму (3НФ).

Приклад виконання

Цей набір відношень знаходиться у 2НФ, але не в 3НФ, оскільки в результаті аналізу початкового відношення була виявлена ще одна функціональна залежність {**Банк виробника**} -> {**МФО банка виробника**}. Крім того відношення "Банківські реквізити виробників" містить транзитивну залежність {**Виробник**} -> {**Банк виробника**} -> {**МФО банку виробника**}.

Для перетворення відношення у 3НФ необхідно позбавитися від виявленої транзитивної залежності. Для цього відношення **"Банківські реквізити виробників"** декомпозуємо на два відношення внаслідок чого початкове відношення опиниться у 3НФ (табл. 4.7).

Таблиця 4.7

Функціональні залежності у 3НФ

R	ПК	Залежні атрибути
"Закупівлі" (Виробник, Товар, Реалізатор, Дата поставки, Кількість)	ПК[Виробник, Товар, Реалізатор, Дата поставки]	Кількість
"Договори на постачання" (Виробник, Реалізатор, № договору)	ПК[Виробник, Реалізатор]	№ договору
"Стандарти на товари" (Товар, ДСТУ на товар)	ПК[Товар, ДСТУ на товар]	ДСТУ на товар
"Розрахункові рахунки виробників" (Виробник, Р/Р виробника в банку, Банк виробника)	ПК[Виробник]	Р/Р виробника в банку, Банк виробника
"МФО банків" (Банк виробника, МФО банку виробника)	ПК[Банк виробника]	МФО банку виробника

Отримані відношення в цьому випадку будуть мати такий вигляд (табл. 4.8 – 4.12):

Таблиця 4.8

Закупівлі

Виробник	Реалізатор	Товар	Дата поставки	Кількість
1	2	3	4	5
АТ "Кулиничі"	ТФ "Барс"	Дарницький	10.10.21	200
АТ "Кулиничі"	ТФ "Булка"	Нарізний	10.10.21	250
ЗАТ "Хліб"	ТФ "Булка"	Нарізний	10.10.21	350
ЗАТ "Хліб"	ТФ "Булка"	Сімейка	10.10.21	200
ЗАТ "Хліб"	ТФ "Здоба"	Сімейка	10.10.21	350
ТОВ "Салтівський"	ТФ" Здоба"	Білий	10.10.21	245
АТ "Кулиничі"	ТФ "Барс"	Нарізний	11.10.21	500
ЗАТ "Хліб"	ТФ "Булка"	Український	11.10.21	150
ТОВ "Салтівський"	ТФ" Здоба	Житній	11.10.21	200
ТОВ "Салтівський"	ТФ" Булка"	Завиток	12.10.21	300

Закінчення табл. 4.8

1	2	3	4	5
ЗАТ "Хліб"	ТФ "Здоба"	Сімейка	12.10.21	120
ТОВ "Салтівський"	ТФ" Здоба"	Завиток	12.10.21	310

Таблиця 4.9

Договори на постачання

Виробник	Реалізатор	№ договору
АТ "Кулиничі"	ТФ "Барс"	12-091
АТ "Кулиничі"	ТФ "Булка"	045-11
ЗАТ "Хліб"	ТФ "Булка"	23-011
ЗАТ "Хліб"	ТФ "Здоба"	114-01
ТОВ "Салтівський"	ТФ" Булка"	045-13
ТОВ "Салтівський"	ТФ "Здоба"	14-235

Таблиця 4.10

Стандарти на товари

Товар	ДСТУ на товар
Білий	4583
Дарницький	4583
Завиток	4585
Нарізний	4583
Житній	4583
Сімейка	4585
Український	4583

Таблиця 4.11

Розрахункові рахунки виробників

Виробник	Р/Р виробника	Банк виробника
АТ "Кулиничі"	26000001	Синтез
ЗАТ "Хліб"	26010234	Ексімбанк
ТОВ "Салтівський	26000234	Синтез

Таблиця 4.12

МФО банків

Банк виробника	МФО банка виробника
Синтез	851300
Ексімбанк	851301

4.2. Операції реляційної алгебри

Завдання

На основі отриманих відношень у 3НФ запишіть мовою реляційної алгебри операції проекції і з'єднання. Кожна операція повинна містити значення вихідних відношень та отримати результівне відношення.

Приклад виконання

Проілюструйте приклади виконання операцій (проекції і з'єднання) на відношеннях предметної області: закупівлі хлібобулочних виробів. Нехай є відношення, що показані у табл. 4.13, 4.14.

Таблиця 4.13

Закупівлі_1

Виробник	Реалізатор	Товар	Дата постачання	Кількість
АТ "Кулиничі"	ТФ "Барс"	Дарницький	10.10.21	200
ЗАТ "Хліб"	ТФ "Булка"	Сімейка	10.10.21	200
ЗАТ "Хліб"	ТФ "Здоба"	Сімейка	10.10.21	350
ТОВ "Салтівський"	ТФ "Здоба "	Білий	10.10.21	245
ТОВ "Салтівський"	ТФ "Булка "	Завиток	12.10.21	300
ЗАТ "Хліб"	ТФ "Здоба"	Сімейка	12.10.21	120
ТОВ "Салтівський"	ТФ "Здоба "	Завиток	12.10.21	310

Таблиця 4.14

Закупівлі _2

Виробник	Реалізатор	Товар	Дата постачання	Кількість
АТ "Кулиничі"	ТФ "Барс"	Дарницький	10.10.21	200
ТОВ "Салтівський"	ТФ "Здоба "	Білий	10.10.21	245
АТ "Кулиничі"	ТФ "Барс"	Нарізний	11.10.21	500
ЗАТ "Хліб"	ТФ "Булка"	Український	11.10.21	150
ТОВ "Салтівський"	ТФ "Здоба "	Житній	11.10.21	200
ТОВ "Салтівський"	ТФ "Булка "	Завиток	12.10.21	300

4.2.1. Операції проекції

Проекція, застосована до відношення "Закупівлі_1", у результаті дає нове відношення, що містить усі кортежі початкового відношення, але тільки певні атрибути, які були визначені в операції. Можна сказати, що це "вертикальна" підмножина початкового відношення.

Результатом операції проекція REZ = $\Pi_{1,3,4}$ ("Закупівлі_1") буде відношення "Закупівлі_ПРЦ" (табл. 4.15).

Таблиця 4.15

Закупівлі_ПРЦ

Виробник	Товар	Дата постачання
АТ "Кулиничі"	Дарницький	10.10.21
ЗАТ "Хліб"	Сімейка	10.10.21
ТОВ "Салтівський"	Білий	10.10.21
ТОВ "Салтівський"	Завиток	12.10.21
ЗАТ "Хліб"	Сімейка	12.10.21

Кортежі результівного відношення, у яких атрибут **Дата постачання** позначений жовтим кольором, дублювалися, проте згідно з вимогами, що висуваються до реляційної моделі, дублювання має бути усунуто.

4.2.2. Операції з'єднання

Для ілюстрації операції з'єднання введіть в розгляд відношення "Розрахункові рахунки виробників" (табл. 4.16).

Таблиця 4.16

Розрахункові рахунки виробників

Виробник	Р/Р виробника	Банк виробника
АТ "Кулиничі"	26000001	Синтез
ЗАТ "Хліб"	26010234	Ексімбанк
ТОВ "Салтівський"	26000234	Синтез

З'єднання відношення "Закупівлі_1" і відношення "Розрахункові рахунки виробників" за загальним атрибутом **Виробник**, дає нове відношення "Закупівлі_ЗДН", кортежі якого є зчепленням двох кортежів (що належать, відповідно, "Закупівлі_1" і "Розрахункові рахунки виробників"), та мають однакове значення загального атрибуту **Виробник**. Причому ці загальні значення в результівному відношенні з'являються тільки один раз.

Можна сказати, що кортежі, які мають однакові значення загальних атрибутів "склеюються". Тоді відношення "Закупівлі_ЗДН", що є результатом виконання операції REZ = "Закупівлі_1" JOIN "Розрахункові рахунки виробників", буде мати вигляд табл. 4.17.

Таблиця 4.17

Закупівлі_ЗДН

Виробник	Реалізатор	Товар	Дата постачання	Кількість	Р/Р виробника	Банк виробника
АТ "Кулиничі"	ТФ "Барс"	Дарницький	10.10.21	200	26000001	Синтез
ЗАТ "Хліб"	ТФ "Булка"	Сімейка	10.10.21	200	26010234	Ексімбанк
ЗАТ "Хліб"	ТФ "Здоба"	Сімейка	10.10.21	350	26010234	Ексімбанк
ТОВ "Салтівський"	ТФ "Здоба"	Білий	10.10.21	245	26000234	Синтез
ТОВ "Салтівський"	ТФ "Булка"	Завиток	12.10.21	300	26000234	Синтез
ЗАТ "Хліб"	ТФ "Здоба"	Сімейка	12.10.21	120	26010234	Ексімбанк
ТОВ "Салтівський"	ТФ "Здоба"	Завиток	12.10.21	310	26000234	Синтез

4.3. Старші нормальні форми

4.3.1. Альтернативні ключі

Завдання

Змініть відношення так, щоб в таблиці з'явилися перекриваючі альтернативні ключі (в разі необхідності введіть додаткові атрибути та сформулюйте додаткові обмеження).

Приклад виконання

Отриманий набір відношень (див. табл. 4.8 – 4.12) знаходиться у 3НФ. Він також знаходиться і нормальній формі Бойса–Кодда (НФБК), оскільки в отриманих відношеннях немає ключів, що перекриваються.

Припустимо, що у відношенні "Договори на постачання" для реалізатора необхідно зберігати код його організації (код ЄДРПОУ). Значення цього коду є унікальним для будь-якої організації (табл. 4.18).

Таблиця 4.18

Модифіковані договори на постачання

Виробник	Реалізатор	Код ЄДРПОУ реалізатора	№ договору
АТ "Кулиничі"	ТФ "Барс"	23476092	12-091
АТ "Кулиничі"	ТФ "Булка"	31512733	045-11
ЗАТ "Хліб"	ТФ "Булка"	31512733	23-011
ЗАТ "Хліб"	ТФ "Здоба"	30011211	114-01
ТОВ "Салтівський"	ТФ" Булка"	31512733	045-13
ТОВ "Салтівський"	ТФ" Здоба"	30011211	14-235

Тоді відношення "Договори на придбання" буде містити такі функціональні залежності (табл. 4.19).

Таблиця 4.19

Договори на придбання

R	X	ФЗ	Y
R5	{Виробник, Реалізатор}	->	{№ Договору }
R6	{Виробник, Код ЄДРПОУ реалізатора }	->	{№ Договору }
R7	{Реалізатор}	->	{Код ЄДРПОУ реалізатора}
R8	{Код ЄДРПОУ реалізатора }	->	{Реалізатор}

Можливими ключами відношення в цьому разі будуть:

[**Виробник, Реалізатор**];

[**Виробник, Код ЄДРПОУ реалізатора**].

Ключі перекриваються, і, який би ключ не був обраний в якості первинного, буде існувати функціональна залежність частини первинного ключа від неключових атрибутів (або {**Реалізатор**} -> {**Код ЄДРПОУ реалізаторів**}, або {**Код ЄДРПОУ реалізаторів**} -> {**Реалізатор**}).

4.3.2. Нормальна форма Бойса–Кодда

Завдання

На основі нових даних, подайте відношення в НФБК.

Приклад виконання

Якщо відношення не перебуває у НФБК, його слід декомпозувати на два відношення, як показано у табл. 4.20, табл. 4.21.

Таблиця 4.20

Договори на постачання_1

Виробник	Код ЄДРПОУ реалізатора	№ договору
АТ "Кулиничі"	23476092	12-091
АТ "Кулиничі"	31512733	045-11
ЗАТ "Хліб"	31512733	23-011
ЗАТ "Хліб"	30011211	114-01
ТОВ "Салтівський"	31512733	045-13
ТОВ "Салтівський"	30011211	14-235

Таблиця 4.21

Реалізатори

Код ЄДРПОУ реалізатора	Реалізатор
23476092	ТФ "Барс"
31512733	ТФ "Булка"
30011211	ТФ "Здоба"

Примітка: можливе інше розбиття початкового відношення.

4.3.3. Перевірка третьої нормальної форми

Завдання

Визначте, чи знаходиться отримане відношення в 3НФ.

Приклад виконання

Отримані відношення знаходяться:

у 2НФ, оскільки всі неключові атрибути неприведено залежать від первинного ключа;

у 3НФ, оскільки в них відсутні транзитивні залежності;

у НФБК, унаслідок того, що усунені ключі, що перекриваються, і відсутні функціональні залежності частини первинного ключа від неключових атрибутів.

4.3.4. Багатозначні залежності предметної області

Завдання

Визначте, чи існують в предметній області багатозначні залежності (БЗ). Якщо їх немає, то сформулюйте обмеження до системи таким чином, щоб в ній були присутні БЗ.

Приклад виконання

Припустимо, що кожен реалізатор має декілька кіосків для реалізації хлібобулочних виробів і парк автомобілів, для доставки продукції від виробників до місця реалізації. Тоді відношення "Реалізатори" можна подати у вигляді відношення "Відомості про реалізаторів" (табл. 4.22).

Таблиця 4.22

Відомості про реалізаторів

Код ЄДРПОУ реалізатора	Реалізатор	Кіоск реалізації	Автотранспорт
23476092	ТФ "Барс"	Кіоск "Вогник"	АХ-1 827
23476092	ТФ "Барс"	Кіоск "Вогник"	АХ-2256
23476092	ТФ "Барс"	Кіоск "Завиток"	АХ-1827
23476092	ТФ "Барс"	Кіоск "Завиток"	АХ-2256
23476092	ТФ "Барс"	Кіоск "Хліб"	АХ-1827
23476092	ТФ "Барс"	Кіоск "Хліб"	АХ-2256
31512733	ТФ "Булка"	Кіоск "Бублик"	ХА-1515
31512733	ТФ "Булка"	Кіоск "Бублик"	АХ-0011
31512733	ТФ "Булка"	Кіоск "Булочки"	ХА-1515
31512733	ТФ "Булка"	Кіоск "Булочки"	АХ-0011
30011211	ТФ "Здоба"	Кіоск "Світанок"	ХА-2345
30011211	ТФ "Здоба"	Кіоск "Світанок"	ХА-4576
30011211	ТФ "Здоба"	Кіоск "Хліб"	ХА-2345
30011211	ТФ "Здоба"	Кіоск "Хліб"	ХА-4576
30011211	ТФ "Здоба"	Кіоск "Печиво"	ХА-2345
30011211	ТФ "Здоба"	Кіоск "Печиво"	ХА-4576

У цьому відношенні існують функціональні залежності (ФЗ) і багатозначні залежності (БЗ), показані в табл. 4.23, 4.24.

Таблиця 4.23

Функціональні залежності

R	X	ФЗ	Y
R9	{Код ЄДРПОУ реалізатора}	->	{Реалізатор}
R10	{Реалізатор}	->	{Код ЄДРПОУ реалізатора}

Таблиця 4.24

Багатозначні залежності

R	X	БЗ	Y
R11	{Реалізатор}	->>	{Кіоски реалізації} | {Автотранспорт}
R12	{Код ЄДРПОУ реалізатора}	->>	{Кіоски реалізації} | {Автотранспорт}

4.3.5. Четверта нормальна форма

Завдання

На основі аналізу БЗ декомпозуйте вихідне відношення так, щоб воно знаходилося в 4НФ.

Приклад виконання

Відношення "Відомості про реалізаторів" не знаходиться в 3НФ. Оскільки атрибути **Кіоск реалізації** та **Автотранспорт** залежать від атрибута **Код ЄДРПОУ реалізатора**, вони транзитивні через атрибут **Реалізатор**, а не напряму.

Виконаємо декомпозицію.

На першому кроці, для декомпозиції відношення "Відомості про реалізаторів" скористаємося теоремою Хеза. Оскільки існує функціональна залежність

R_9 = {**Код ЄДРПОУ реалізатора**} -> {**Реалізатор**},

розіб'ємо відношення на два відношення (табл. 4.25).

Таблиця 4.25

Залежний атрибут Реалізатор

R	ПК	Залежний атрибут
"Реалізатори" (Код ЄДРПОУ реалізатора, Реалізатор)	ПК[Код ЄДРПОУ реалізатора]	Реалізатор
"Доставки реалізатора" (Код ЕДРПОУ реалізатора, Кіоск реалізації, Автотранспорт)	ПК[Код ЕДРПОУ реалізатора, Кіоск реалізації, Автотранспорт]	

Створюється декомпозиція без втрат. Можна з'єднати отримані два відношення, отримавши початкове.

Отримана декомпозиція залишає БЗ у відношенні "Доставка реалізаторів", яку можна усунути, скориставшись теоремою Фейгіна, розбивши його на два відношення (табл. 4.26).

Таким чином, приведення початкового відношення "Відомості про реалізаторів" до 4НФ з метою усунення багатозначних залежностей дозволило отримати нові відношення (табл. 4.27 – 4.29).

Таблиця 4.26

Доставка реалізаторів

R	ПК	Залежний атрибут
"Кіоски реалізаторів" (Код ЄДРПОУ реалізатора, Кіоск реалізації)	ПК[Код ЄДРПОУ реалізатора, Кіоск реалізації]	
"Автотранспорт реалізаторів" (Код ЄДРПОУ реалізатора, Автотранспорт)	ПК[Код ЕДРПОУ реалізатора, Автотранспорт]	

Таблиця 4.27

Реалізатори (до 4НФ)

Код ЄДРПОУ реалізатора	Реалізатор
23476092	ТФ "Барс"
31512733	ТФ "Булка"
30011211	ТФ "Здоба"

Таблиця 4.28

Кіоски реалізаторів (до 4НФ)

Код ЄДРПОУ реалізатора	Кіоск реалізації
23476092	Кіоск "Вогник"
23476092	Кіоск "Завиток"
23476092	Кіоск "Хліб"
31512733	Кіоск "Бублик"
31512733	Кіоск "Булочки"
30011211	Кіоск "Світанок"
30011211	Кіоск "Хліб"
30011211	Кіоск "Печиво"

Таблиця 4.29

Автотранспорт реалізаторів (до 4НФ)

Код ЄДРПОУ реалізатора	Автотранспорт
23476092	АХ-1827
23476092	АХ-2256
31512733	ХА-1515
31512733	АХ-0011
30011211	ХА-2345
30011211	ХА-4576"

Таким чином, у результаті проведених операцій початкове відношення "Постачання хлібобулочних виробів " було нормалізоване до третьої нормальної форми.

Крім того були проілюстровані необхідні кроки, які дозволять перевести відношення у нормальну форму Бойса-Кодда та четверту нормальну форму.

Зауважимо, що для практичного проектування достатньо 3НФ.

4.3.6. П'ята нормальна форма

Завдання

Перевірте, чи можна на деякій підмножині атрибутів предметної області побудувати відношення, для якого б виконувалася залежність з'єднання (5НФ).

Приклад виконання

Припустимо, що потрібно зберігати дані про асортимент реалізаторів, які торгують продукцією виробників (номенклатура товарів виробників може перетинатися) (табл. 4.30).

Таблиця 4.30

Асортимент

Реалізатор	Виробник	Товар
ТФ "Барс"	АТ "Кулиничі"	Дарницький
ТФ "Булка"	АТ "Кулиничі"	Нарізаний
ТФ "Булка"	ЗАТ "Хліб"	Нарізаний
ТФ "Булка"	ЗАТ "Хліб"	Сімейка
ТФ "Здоба"	ЗАТ "Хліб"	Сімейка
ТФ" Здоба"	ТОВ "Салтівський"	Білий
ТФ "Барс"	АТ "Кулиничі"	Нарізаний
ТФ "Булка"	ЗАТ "Хліб"	Український
ТФ" Здоба"	ТОВ "Салтівський"	Житній
ТФ "Булка"	ТОВ "Салтівський"	Завиток
ТФ "Здоба"	ЗАТ "Хліб"	Сімейка
ТФ "Здоба "	ТОВ "Салтівський"	Завиток

Якщо додаткових умов немає, то відношення, яке знаходиться в 4НФ, є коректним і відображає всі необхідні обмеження.

Тепер припустимо, що потрібно врахувати таке обмеження: кожен реалізатор має в своєму асортименті обмежений список виробників і обмежений

список типів товарів. Реалізатор пропонує товари зі списку тих, що виробляються виробниками. Тобто реалізатор не має права торгувати якими завгодно товарами яких завгодно виробників.

Таке обмеження може бути викликано, наприклад, тим, що список типів товарів реалізатора обмежений наявними у нього ліцензіями.

Запропоноване вище відношення не може виключити ситуації, за якої це обмеження буде порушено.

Відношення "Асортимент" не перебуває у 5НФ, оскільки в ньому є нетривіальна залежність з'єднання (табл. 4.31).

Таблиця 4.31

Нетривіальні залежності

R	X	ФЗ	Y
R13	{Реалізатор}	->	{Виробник}
R14	{Виробник}	->	{Товар}
R15	{Реалізатор}	->	{Товар}

Проте підмножини {**Реалізатор**, **Виробник**}, {**Виробник**, **Товар**}, {**Реалізатор**, **Товар**} не є суперключами вихідного відношення.

У цьому разі для приведення до 5НФ відношення має бути розбите на три відношення (табл. 4.32 – 4.34).

Таблиця 4.32

Реалізатори_Виробники

Реалізатор	Виробник
ТФ "Барс"	АТ "Кулиничі"
ТФ "Булка"	АТ "Кулиничі"
ТФ "Булка"	ЗАТ "Хліб"
ТФ "Булка"	ТОВ "Салтівський"
ТФ "Здоба"	ЗАТ "Хліб"
ТФ" Здоба"	ТОВ "Салтівський"

Таблиця 4.33

Виробники_Товари

Виробник	Товар
1	2
АТ "Кулиничі"	Дарницький
АТ "Кулиничі"	Нарізаний
ТОВ "Салтівський"	Білий

Закінчення табл. 4.33

1	2
ТОВ "Салтівський"	Житній
ТОВ "Салтівський"	Завиток
ЗАТ "Хліб"	Український
ЗАТ "Хліб"	Нарізаний
ЗАТ "Хліб"	Сімейка

Таблиця 4.34

Реалізатори_Товари

Реалізатор	Товар
ТФ "Барс"	Дарницький
ТФ "Булка"	Нарізаний
ТФ "Булка"	Нарізаний
ТФ "Булка"	Сімейка
ТФ "Здоба"	Сімейка
ТФ "Здоба"	Білий
ТФ "Барс"	Нарізаний
ТФ "Булка"	Український
ТФ "Здоба"	Житній
ТФ "Булка"	Завиток
ТФ "Здоба"	Сімейка
ТФ "Здоба"	Завиток

4.4. Денормалізація

Завдання

Проілюструйте виконання основних операцій денормалізації (дублювання, попередня підготовка) на прикладі відношень з предметної області.

Приклад виконання

База даних "Постачання хлібобулочних виробів", яка знаходиться в 2НФ містить відношення: "Закупівлі" (**Виробник, Товар, Реалізатор, Дата постачання, Кількість**), "Договори на постачання" (**Виробник, Реалізатор, № договору**), "Стандарти на товари" (**Товар, ДСТУ на товар**), "Банківські реквізити виробників" (**Виробник, Р/Р виробника в банку, Банк виробника, МФО банку виробника**). Модифікуємо відношення "Закупівлі" у відношення "Закупівлі_М" додаванням атрибутів: **№ договору, Ціна закупівлі, Сума закупівлі** (табл. 4.35).

Таблиця 4.35

Закупівлі_М

Виробник	Реалізатор	Товар	Дата закупівлі	№ договору	Кількість	Ціна закупівлі	Сума закупівлі
АТ "Кулиничі"	ТФ "Барс"	Дарницький	10.10.21	12-091	200	10,00	2000,00
АТ "Кулиничі"	ТФ "Булка"	Нарізний	10.10.21	045-11	250	8,00	2000,00
ЗАТ "Хліб"	ТФ "Булка"	Нарізний	10.10.21	23-011	350	6,00	2100,00
ЗАТ "Хліб"	ТФ "Булка"	Сімейка	10.10.21	23-011	200	10,00	2000,00
ЗАТ "Хліб"	ТФ "Здоба"	Сімейка	10.10.21	114-01	350	9,00	3150,00
ТОВ "Салтівський"	ТФ" Здоба"	Білий	10.10.21	045-13	245	8,00	1960,00
АТ "Кулиничі"	ТФ "Барс"	Нарізний	11.10.21	12-091	500	6,00	3000,00
ЗАТ "Хліб"	ТФ "Булка"	Український	11.10.21	23-011	150	9,00	1350,00
ТОВ "Салтівський"	ТФ" Здоба	Житній	11.10.21	14-235	200	8,00	1600,00
ТОВ "Салтівський"	ТФ" Булка"	Завиток	12.10.21	045-13	300	5,00	1500,00
ЗАТ "Хліб"	ТФ "Здоба"	Сімейка	12.10.21	114-01	120	8,00	960,00
ТОВ "Салтівський"	ТФ" Здоба"	Завиток	12.10.21	14-235	310	8,00	2480,00

З метою розв'язання завдання слід дотримуватися такої послідовності дій.

Дублювання. У відношенні "Закупівлі_М" присутній атрибут **№ договору** з іншого відношення "Договори на постачання", що дозволяє отримати дані з таблиці "Закупівлі_М" відразу з номером договору за один простий запит вибірки. Наприклад, така швидка видача результату корисна у ході перевірки з боку податкових служб.

Попередня підготовка. Агрегатні запити зазвичай найбільш повільні по своїй продуктивності. Наприклад, отримання кількості записів за певних умов.

Крім дублювання даних з одних таблиць в інші, можна також зберігати дані, які розраховуються. Тоді можна буде уникнути повільних агрегатних вибірок. Наприклад, у таблиці "Закупівлі_М" для зберігання суми закупівлі товару є додаткова колонка (поле що обчислюється: **Сума закупівлі** = **Ціна закупівлі** * **Кількість**).

У результаті можна вибрати дані з таблиці "Закупівлі_М" відразу із сумою закупівлі товару без з'єднання з іншою таблицею за один простий запит вибірки.

Завдання для самостійного виконання

1. На основі отриманих відношень у 3НФ запропонуйте схему БД, що доречна для реалізації в СУБД.

2. На основі відношень у 3НФ (табл. 4.13, 4.14) запишіть мовою реляційної алгебри операцію вибірки.

3. На основі відношень у 3НФ (табл. 4.13, 4.14) запишіть мовою реляційної алгебри операцію об'єднання.

4. На основі відношень у 3НФ (табл. 4.13, 4.14) запишіть мовою реляційної алгебри операцію перетину.

5. На основі відношень у 3НФ (табл. 4.13, 4.14) запишіть мовою реляційної алгебри операцію віднімання.

6. На основі відношень у 3НФ (табл. 4.13, 4.14) запишіть мовою реляційної алгебри операцію декартового добутку.

7. На основі відношень у 3НФ (табл. 4.13, 4.14) запишіть мовою реляційної алгебри операцію ділення.

Примітка: предметна область за варіантами показана у табл. 4.36.

Таблиця 4.36

Варіанти предметних областей

Варіант	Предметна область
1	2
1	Телефонний довідник
2	Спеціалізована бібліотека (книжковий фонд)
3	Спеціалізована бібліотека (облік видачі та повернення книг)
4	Видавництво (облік замовлень)
5	Видавництво (облік випущеної друкованої продукції)
6	Комерційна стоматологічна поліклініка (облік роботи фахівців)
7	Комерційна стоматологічна поліклініка (облік пацієнтів стоматологічної поліклініки)
8	Ательє мод (облік роботи закрійників)
9	Ательє мод (облік роботи складу в ательє мод)
10	Ательє мод (облік замовлень на індпошиття одягу)
11	Оптовий склад (облік товарів)
12	Оптовий склад (реалізація товарів)
13	Оптовий склад (надходження товарів)
14	Торгово-закупівельне підприємство (облік продажу реалізаторами)
15	Торгово-закупівельне підприємство (облік товарів, виданих реалізаторами)
16	Автосалон (технічні характеристики автомобілів)
17	Автосалон (облік відомостей про реалізацію автомобілів)
18	Продаж автомобілів
19	Асоціація селянських фермерських господарств
20	Пасажирське автопідприємство (маршрути)

Закінчення табл. 4.36

1	2
21	Пасажирське автопідприємство (обсяг перевезень)
22	Пасажирське автопідприємство (автопарк)
23	Міжміські пасажирські перевезення (технічні характеристики маршрутів)
24	Міжміські пасажирські перевезення (перевезення пасажирів)
25	Агентство з продажу авіаквитків

Приклади опису предметної області

II. Спеціалізована бібліотека (книжковий фонд)

Спеціалізована бібліотека має в своєму розпорядженні книжковий фонд певної тематичної спрямованості. Передбачається, що кожна книга фонду може бути як в одному екземплярі, так і в декількох. Тому кожній книзі відповідає унікальний інвентарний номер і бібліотечний код. Дані про книги містяться в бібліографічній картці. Картки об'єднуються в каталоги. Існує два види каталогів: алфавітний і тематичний. В алфавітному каталозі картки відсортовані за прізвищем автора, а в тематичному – спочатку по темам, а в межах кожної теми – по прізвищу автора.

Бібліотека видає книги читачам у тимчасове користування. При запису до бібліотеки кожному читачеві присвоюється порядковий номер. Йому видається читацький квиток і для нього заводиться облікова картка. Облікова картка крім даних про читача в подальшому буде містити інформацію про видані та повернуті книги.

Інформація, що характеризує роботу бібліотеки з книгами і читачами, може містити дані про: інвентарний номері книги, бібліотечний код книги, відмітку про видачу/повернення книги, автора і назву книги, видавництво, рік видання, кількість сторінок, теми, ціну, номер читацького квитка, прізвище ім'я та по батькові читача, домашню адресу читача, домашній та мобільний телефон і т. д.

Інформаційна система призначена, перш за все, для ведення даних по книжковому фонду.

Примітка: інші приклади опису предметних областей можна взяти з [2].

Лабораторна робота 5
Побудова логічної і фізичної моделі бази даних CASE-засобами

Цілі роботи

1. Вивчити методологію IDEF1X.
2. Освоїти інструментарій ERwin.
3. Ознайомитися з технологією побудови логічної моделі в ERwin.
4. Вивчити методи визначення ключових атрибутів сутностей.
5. Освоїти метод перевірки адекватності логічної моделі.
6. Вивчити типи зв'язків між сутностями.
7. Вивчити види нормальних форм.
8. Освоїти роль CASE-засобу ERwin у нормалізації і денормалізації бази даних.
9. Побудувати фізичну модель.
10. Вивчити види звітів (для самостійного вивчення).
11. Освоїти процедуру створення звітів (для самостійного опрацювання).
12. Вивчити експортування, збереження і друкування звітів (для самостійного опрацювання).

Перед виконанням роботи студент повинен знати:

поняття семантичного моделювання предметної області;

основні конструктивні елементи ER-моделі;

поняття ключа і їх види;

типи зв'язків, що застосовуються в ER-моделюванні предметної області;

поняття залежної і незалежної сутності;

основні нотації, що використовуються для зображення ER-діаграм.

Після виконання лабораторної роботи студент повинен уміти:

аналізувати предметну область і синтезувати її модель з використанням CASE-технологій;

самостійно формувати прості і складні ER-моделі предметної області;

проводити пряме і зворотне проектування схеми бази даних;

визначати стратегії підтримки посилальної цілісності бази даних;

формувати звіти по сформованій моделі предметної області.

Підготовча частина

Хід роботи

5.1. Підготовчий етап.

5.2. Базовий рівень лабораторної роботи: "Побудова найпростішої ER-моделі і проведення прямого проектування БД".

5.3. Розширений рівень лабораторної роботи: створення моделі предметної області "ІС компанії з продажу товарів".

Форма звітності

За результатами виконання роботи слід оформити електронний звіт засобами Word. В електронному звіті за кожним завданням надати текст завдання з коментарями за формою, що подана в наведених нижче прикладах.

Критерії оцінювання

У 12-бальній системі оцінка результату захисту лабораторної роботи формується за такими правилами:

1. За підготовчий етап і тест з основних понять лабораторної роботи (Інтернет доступ: https://www.it-karkas.com.ua/bd3/indextest.php) може бути виставлено від 0 до 2 балів.

2. За базовий рівень лабораторної роботи може бути виставлено від 0 до 4 балів.

3. За розширений рівень лабораторної роботи може бути виставлено від 0 до 6 балів.

4. За запізнення із захистом лабораторної роботи знімається 2 бали за кожний тиждень.

Рекомендована література: [10; 11; 20].

Основні поняття

Одним з найбільш популярних засобів інформаційного моделювання та автоматизації є CASE-засіб (Computer-Aided Software Engineering) – ERwin Data Modeler фірми Computer Associates. Реалізація моделювання в ERwin базується на теорії реляційних баз даних і на методології IDEF1X.

Логічний (інфологічний) рівень проектування бази даних припускає, що ми мислимо в поняттях реального світу та безпосередньо з нього беремо об'єкти для моделювання.

Фізичний (даталогічний) рівень проектування бази даних передбачає інтерпретацію логічного рівня в термінах фізичної бази даних, включаючи вибір СУБД, типів даних за замовчуванням і ефективних схем індексування.

Діаграма "сутність – зв'язок" ERD (Entity-Relationship Diagram) призначена для розробки моделі даних і забезпечує графічний засіб визначення даних і відношень між ними.

Основними *складовими моделі IDEF1X* є: люди, предмети, явища, про які зберігається інформація (надалі – сутності), зв'язки між цими елементами (надалі – відношення), характеристики цих елементів (надалі – атрибути).

Сутність – це множина реальних або абстрактних об'єктів (людей, місць, подій), яким притаманні загальні атрибути або характеристики. Будь-який об'єкт системи може бути представлений тільки однією сутністю, яка повинна бути унікально ідентифікована.

Відношення – зв'язок між двома та більше сутностями. Іменування відношення здійснюється за допомогою граматичної форми дієслова (має, визначає, ...).

Таким чином: сутності – це базовий тип інформації, що зберігається в базі даних, а відношення показують, як ці типи даних взаємопов'язані між собою. Сутність повинна мати унікальне ім'я і іменуватися іменником в однині.

Сутність має один або кілька атрибутів, які однозначно ідентифікують кожен зразок сутності та називаються ключем (складеним ключем). Кожна сутність може мати будь-яку кількість відношень з іншими сутностями.

Якщо зовнішній ключ цілком використовується в складі первинного ключа, то сутність є залежною від ідентифікатора.

У нотації IDEF1X сутність зображується у вигляді прямокутника. Залежно від рівня подання даних можуть бути деякі відмінності.

Розрізняють такі рівні подання сутності: діаграма "сутність – зв'язок" (ERD); модель даних, заснована на ключах (KB); повна атрибутивна модель (FA). Кожен атрибут кожної сутності має унікальне ім'я. Сутність може мати будь-яку кількість атрибутів.

Розрізняють власні та успадковані **атрибути**. *Власні атрибути* є унікальними в рамках моделі. *Успадковані* передаються від сутності-"батька" шляхом визначення ідентифікованого зв'язку.

Ключовий елемент таблиці (ключ) – таке її поле (простий ключ) або рядкове вираження, утворене зі значень декількох полів (складений ключ), через який можна визначити значення інших полів для однієї або кількох записів таблиці.

На практиці для використання ключів створюються **індекси** – службова інформація, що містить упорядковані відомості про ключові значення. У реляційній теорії і концептуальній моделі поняття "ключ" застосовується для атрибутів відношення або сутності.

Незалежна сутність IDEF1X – це незалежні дані, які завжди присутні в системі. Водночас відношення з іншими сутностями можуть як існувати, так і бути відсутніми.

Залежна сутність IDEF1X подає дані, залежні від інших сутностей в системі; тому вона завжди повинна мати відносини з іншими сутностями.

У методі IDEF1X усі сутності розрізняють на *залежні* та *незалежні від ідентифікаторів*. Сутність є незалежною від ідентифікаторів або просто незалежною, якщо кожен її екземпляр може бути однозначно ідентифікований без визначення його відношень з іншими сутностями. Сутність називають залежною від ідентифікаторів або просто залежною, якщо однозначна ідентифікація примірника сутності залежить від його відношення до іншої сутності. Незалежна сутність зображується у вигляді звичайного прямокутника, залежна – у вигляді прямокутника з закругленими кутами.

У IDEF1X існують такі **види потужностей зв'язків**:

N потужність – кожен екземпляр сутності-батька може мати нуль, один або більше одного пов'язаного з ним екземпляра сутності-нащадка (за замовчуванням);

P потужність – кожен екземпляр сутності-батька повинен мати не менше одного пов'язаного з ним екземпляра сутності-нащадка;

Z потужність – кожен екземпляр сутності-батька повинен мати не більше одного пов'язаного з ним екземпляра сутності-нащадка;

конкретне число – кожен екземпляр сутності-батька пов'язаний з деяким фіксованим числом екземплярів сутності-нащадка.

За замовчуванням потужність зв'язків приймається дорівненою N.

Зв'язок "один-до-одного" означає, що екземпляр однієї сутності пов'язаний тільки з одним екземпляром іншої сутності. Зв'язок 1:M означає, що один екземпляр сутності, розташований ліворуч лінії зв'язка, може бути пов'язаний з декількома екземплярами сутності, розташованими праворуч. Зв'язок "багато-до-багатьох" (M:M) означає, що один екземпляр першої сутності може бути пов'язаний з декількома екземплярами другої сутності, і навпаки, один екземпляр другої сутності може бути пов'язаний з декількома екземплярами першої сутності.

Зв'язок зображується лінією, що проводиться між сутністю-батьком і сутністю-нащадком, з точкою на кінці лінії у сутності-нащадка.

Якщо екземпляр сутності-нащадка однозначно визначається своїм зв'язком з сутністю-батьком, то зв'язок називають *ідентифікуючим*, інакше – *неідентифікуючим*.

Ідентифікуючий зв'язок зображується суцільною лінією, неідентифікуючий – пунктирною лінією.

Атрибути зображуються у вигляді списку імен усередині блоку сутності. Атрибути, що визначають первинний ключ, розміщуються напочатку та відділяються від інших атрибутів горизонтальною рискою.

Сутності можуть мати також зовнішній ключ (Foreign Key), який використовується як або первинний ключ або неключовий атрибут. Для позначення зовнішнього ключа всередині блоку сутності поміщають імена атрибутів, після яких записують букви FK у дужках.

ERwin – CASE-засіб для проектування та документування баз даних, який дозволяє створювати, документувати та супроводжувати бази даних, сховища і вітрини даних (версія Erwin 4.0 Build 1298).

У ERwin'і зі встановленням ідентифікованого зв'язку атрибути первинного ключа батьківської сутності автоматично переносяться до складу первинного ключа дочірньої сутності. Цю операцію називають **міграцією атрибутів**. У дочірньої сутності нові атрибути позначаються як зовнішній ключ (FK). Зі встановленням неідентифікуючого зв'язку атрибути первинного ключа батьківської сутності мігрують до складу неключових полів дочірньої сутності.

Робота з програмою починається зі створення нової моделі, для якої потрібно вказати тип і цільову СУБД (рис. 5.1).

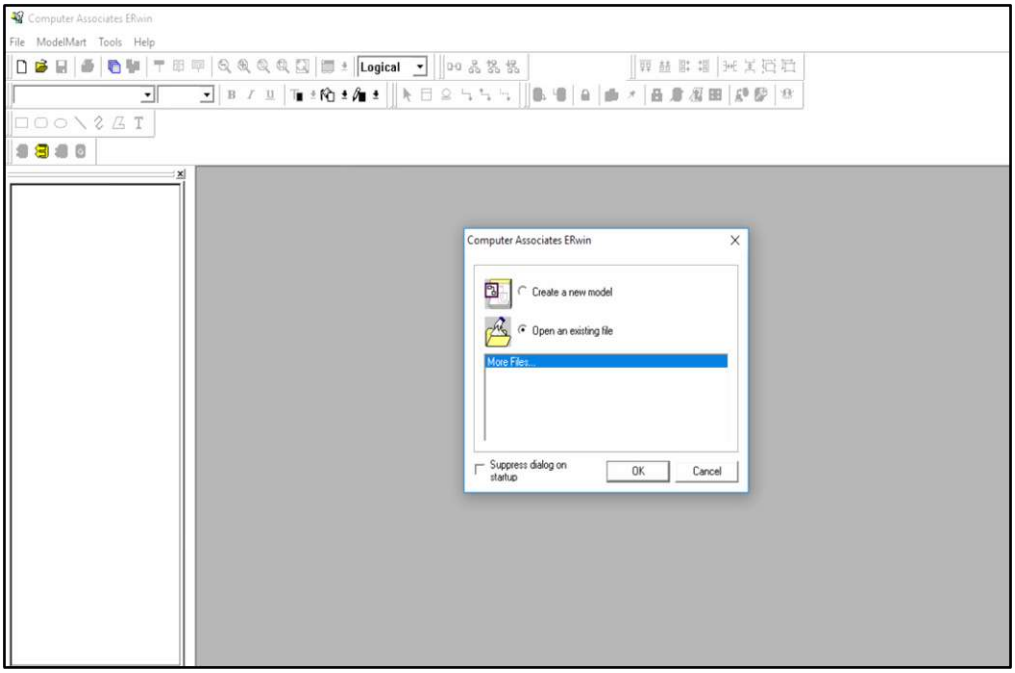


Рис. 5.1. **Початковий вид вікна ERwin**

ERwin дозволяє створювати логічну, фізичну моделі та модель, яка поєднує логічний і фізичний рівні.

Для створення нової моделі необхідно вказати її тип: логічний, фізичний або логічно/фізичний (рис. 5.2).

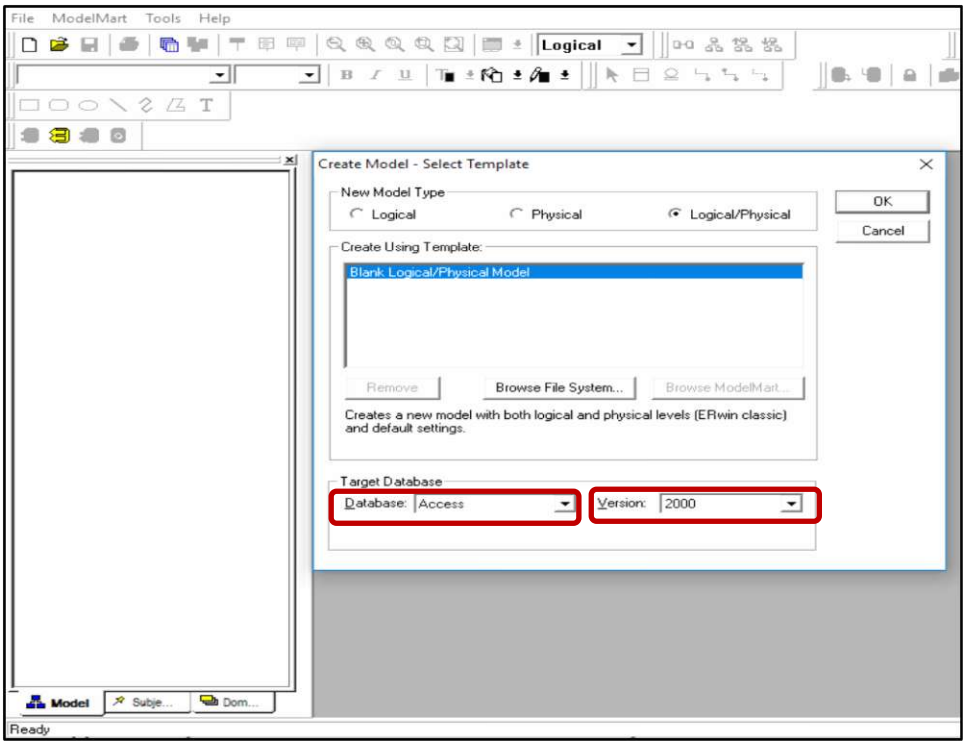


Рис. 5.2. **Вибір типу нової моделі**

Після вибору режиму створення моделі або відкриття вже існуючої моделі, палітра інструментів стає доступною.

Далі буде розглядатися робота з ERwin в нотації IDEF1X.

Примітка. У створеній моделі з налаштуванням за замовчуванням некоректно відображаються українські символи. Щоб усунути цей недолік, необхідно підкоригувати використані в моделі шрифти. Для цього необхідно зайти в меню Format -> Default Fonts & Colors, послідовно пройтися усіма вкладками, вибравши будь-який шрифт, назва якого закінчується на CYR (наприклад, Arial CYR), і виставити перемикач Apply To в значенні All Objects.

Практична частина

5.1. Підготовчий етап

Завдання

Створіть порожню базу даних у СУБД Access 2013.

Примітка:

1) ця база даних буде автоматично заповнюватися порожніми таблицями у п. 5.2;

2) оскільки робоча версія програми ERwin, що використовується в лабораторній роботі, створена давно, вона може співпрацювати з базою даних у форматі Access 2000.

Виконання

1. Створіть порожню базу даних у СУБД Access 2013.

2. Запустіть Access 2013, у правому кутку виберіть **Нова база даних** і клацніть значок **Створити**.

3. Збережіть нову базу як БД Access 2000, потім укажіть ім'я і місце розташування і клацніть **Зберегти**.

5.2. Базовий рівень лабораторної роботи: "Побудова найпростішої ER-моделі та проведення прямого проектування БД"

Спроектувати просту інформаційну систему, яка базується на розглянутій у лабораторній роботі 4 системі обліку продажу хлібобулочних виробів.

5.2.1. Створення моделі

Завдання

Створіть нову модель в ERwin.

Виконання

1. Запустіть Erwin.
2. Виберіть **Create a new model** (Створіть нову модель).
3. Виберіть тип моделі **Logical/Physical**. Це дозволить надалі легко перемикатися між логічним і фізичним рівнями. У списку пропонованих баз даних виберіть Access. Пізніше, у разі потреби, середовище реалізації можна буде змінити.

5.2.2. Створення сутності

Завдання

Створіть сутність ТОВАРИ в ERwin.

Примітка: На фізичному рівні сутність подають таблицею, атрибут – колонкою таблиці, екземпляр – рядком таблиці. Для наведеного прикладу виділіть такі основні логічні об'єкти: **Товари** та **Виробники**.

Виконання

1. Клацніть значок ⊟ на панелі інструментів. Курсор набере вигляду ⊹⊟.
2. Клацніть на будь-якому місці діаграми. На екрані з'явиться нова сутність.
3. За замовчуванням, сутності надається ім'я **E/x**, де x – унікальний номер сутності. Змінити ім'я можна, клацнувши на сутності лівою кнопкою миші або вибравши пункт **Entity Properties** у контекстному меню сутності.
4. У властивостях сутності також можна дати опис сутності, ввести замітки та вказати іншу інформацію.

Примітка. Обов'язково складіть описи створюваних сутностей. Це допоможе надалі зрозуміти, що це за об'єкт. Зробіть схему читаною для інших учасників процесу розроблення. Крім того, побудувавши після створення діаграми звіт, можна отримати готову документацію за схемою.

Для внесення опису та зауважень виберіть в контекстному меню сутності пункт меню **Entity Properties**. У полі **Definition** вкажіть визначення сутності.

На фізичному рівні визначення сутності можна експортувати як частину схеми і використовувати в реальній базі даних.

5.2.3. Створення атрибутів

Завдання

Створіть атрибути для сутності "ТОВАРИ".

Виконання

1. Виділіть сутність.

2. Клацніть правою кнопкою і виберіть **Attributes** в контекстному меню сутності. З'являється діалог **Attributes**.

3. Щоб додати новий атрибут клацніть кнопку **New**, введіть ім'я атрибута, ім'я колонки до бази даних і його домен. Домен атрибута використовується під час визначення типу колонки на фізичному рівні. Для атрибутів первинного ключа у вкладці **General** зробіть позначку у вікні вибору **Primary Key**.

Аналогічно сутностям, необхідно задавати опис атрибутам (вкладка **Definition**). Далі створіть сутність ВИРОБНИКИ й атрибути для неї.

Якщо встановити рівень перегляду атрибутів (**Attribute level**), загальний вид екрану в цьому випадку виглядатиме як на рис 5.3.

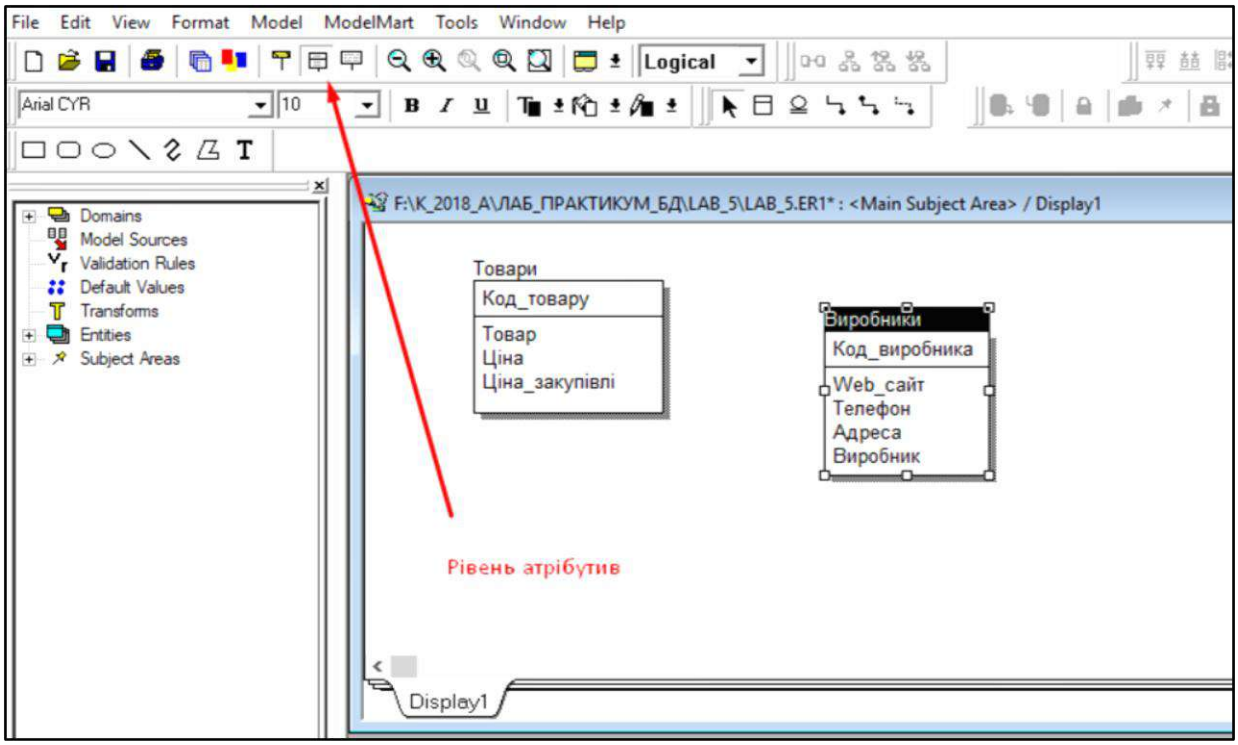


Рис. 5.3. **Вид екрану для двох створених сутностей**

5.2.4. Створення зв'язку

Завдання

Створіть зв'язок "Виробники виробляють Товари".

Виконання

1. Клацніть на одній з кнопок на панелі інструментів.
2. Клацніть на батьківській сутності.
3. Клацніть на дочірній сутності.

Примітка. У цьому прикладі ми не вводимо ідентифікуючих зв'язків. Кожен екземпляр усіх сутностей однозначно визначається своїм номером. У ілюстрованому прикладі зв'язок "Виробники виробляють Товари" матиме зв'язок "багато-до-багатьох", який можна створити тільки на рівні логічної моделі. За замовчуванням дієслівна фраза не відображається на екрані. Для того щоб показати на екрані дієслівну фразу, що належить до зв'язку, клацніть правою кнопкою миші на будь-якому місці у вікні діаграми та виберіть пункт **Relationship Display -> Verb Phrase** (рис. 5.4).

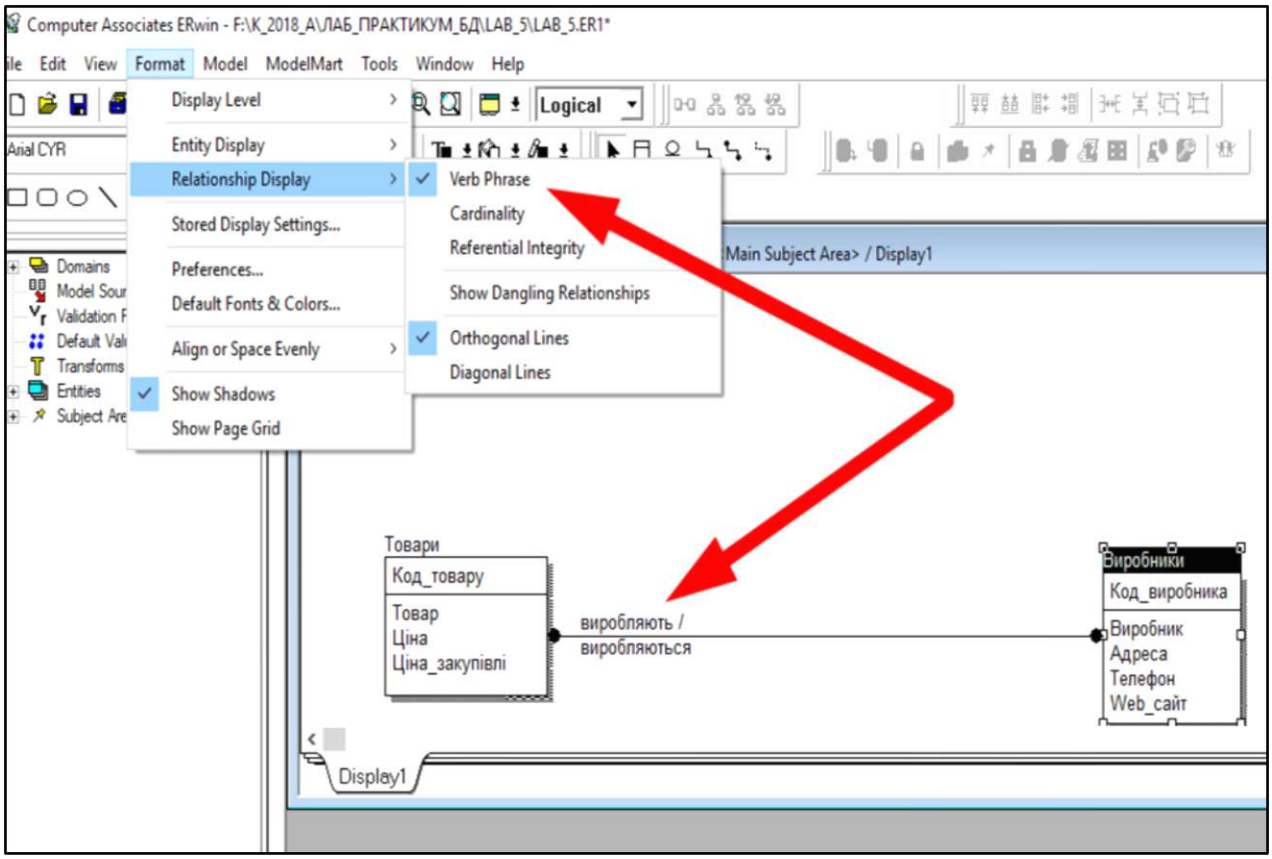


Рис. 5.4. **Відображення зв'язку на діаграмі**

5.2.5. Створення фізичної моделі даних

Завдання

Позбавтесь від зв'язків "багато-до-багатьох".

Виконання

1. Виділіть зв'язок і запустіть майстер перетворення зв'язка "багато-до-багатьох" .

2. Після запуску майстра в його початковому вікні клацніть кнопку **Далі** і перейдіть у вікно **Transform information**, де вкажіть ім'я перетворення та його опис.

3. Клацніть кнопку **Далі**, а потім **Готово**. У результаті діаграма виглядатиме як на рис. 5.5.

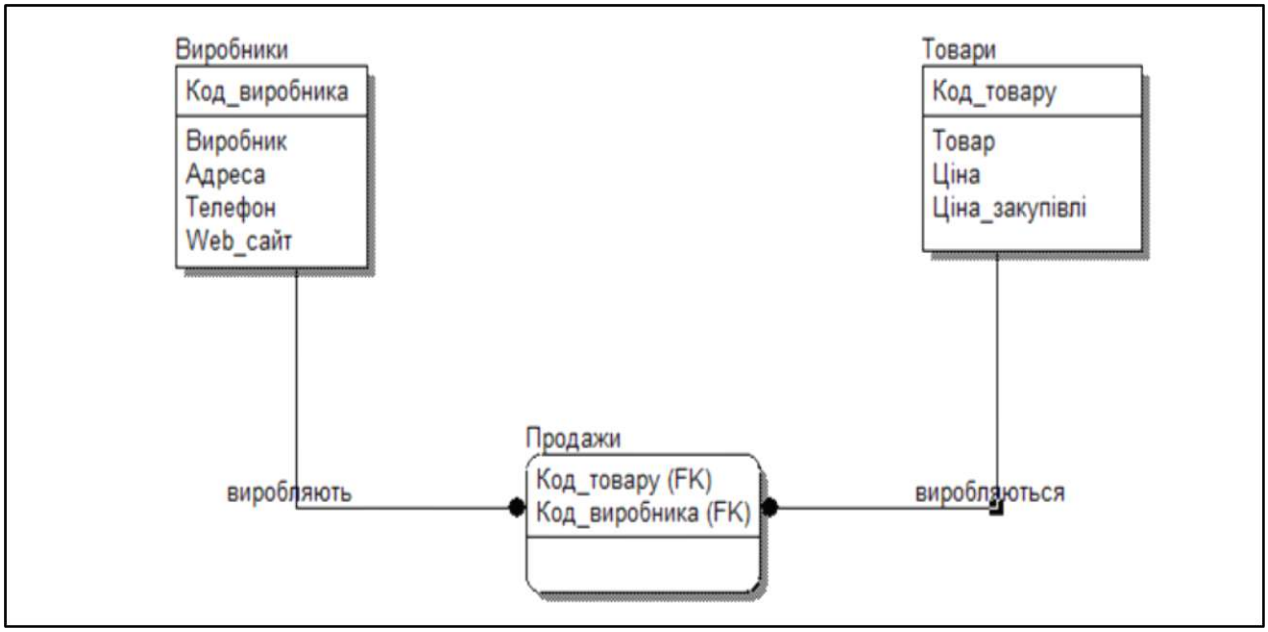


Рис. 5.5. **Вид створеної діаграми після перетворення зв'язку**

Примітка. З'являється нова сутність "ПРОДАЖІ" така, що забезпечує асоціативний зв'язок між виробниками та товарами. Поява такої сутності позбавляє нас від зв'язку "багато-до-багатьох". Первинні ключі початкових сутностей стають складеним первинним ключем нової асоціативної сутності.

4. Після цього додайте до сутності "ПРОДАЖІ" нові атрибути: Дата і Кількість.

5.2.6. Генерація фізичної схеми бази даних

Завдання

Виконайте генерацію фізичної схеми бази даних (Forward Engineering).

Виконання

1. Для генерації системного каталогу бази даних перейдіть з логічної моделі на фізичну, потім виберіть пункт меню **Tools -> Forward Engineer/Schema Generation**.

2. Для генерації схеми бази даних клацніть кнопку **Generate** та увійдіть до діалогу **Connection** для встановлення сеансу зв'язку з сервером і для запуску SQL-скрипта (рис. 5.6).

3. Клацніть кнопку **Connect**. Відбувається запуск процесу генерації таблиць у вибраній базі даних.

4. Відкривши створену базу даних в ACCESS, побачимо схему, що зображена рис. 5.7.

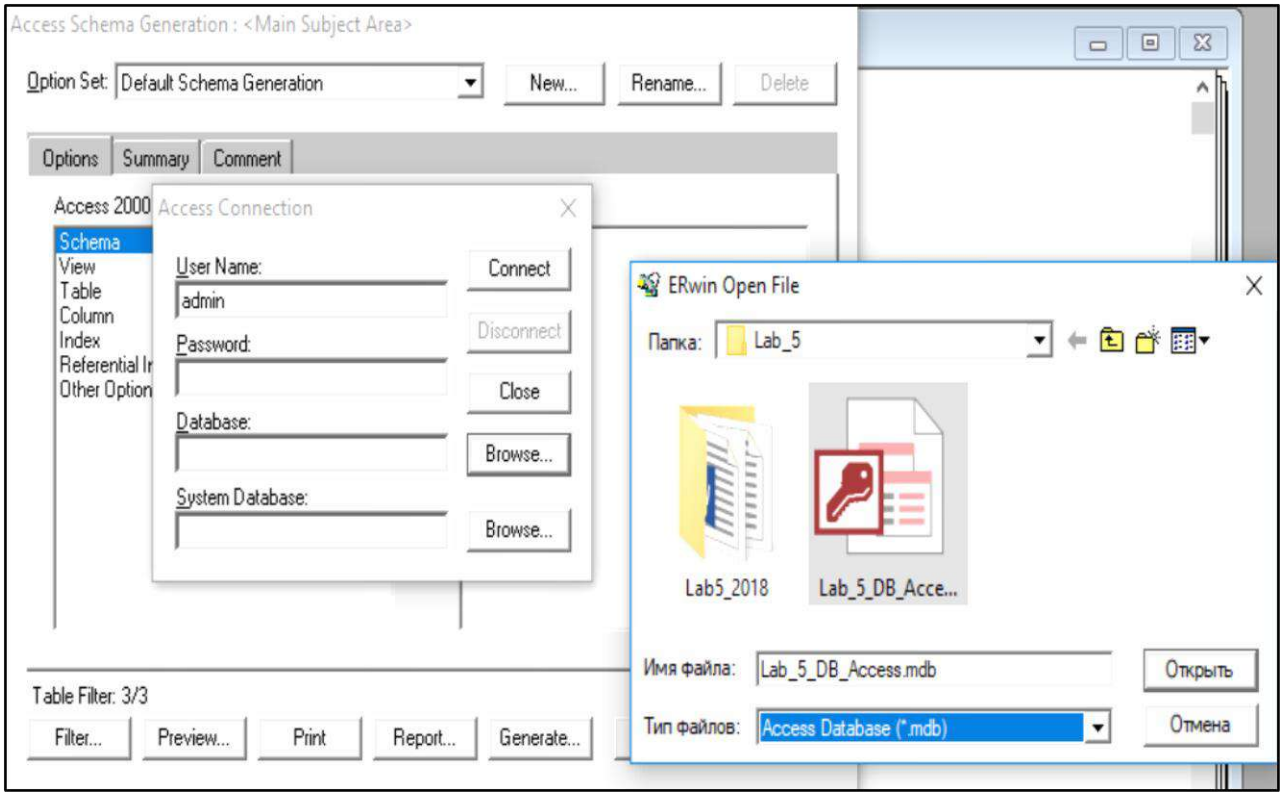


Рис. 5.6. **Вікно вибору бази даних під час прямого проектування**

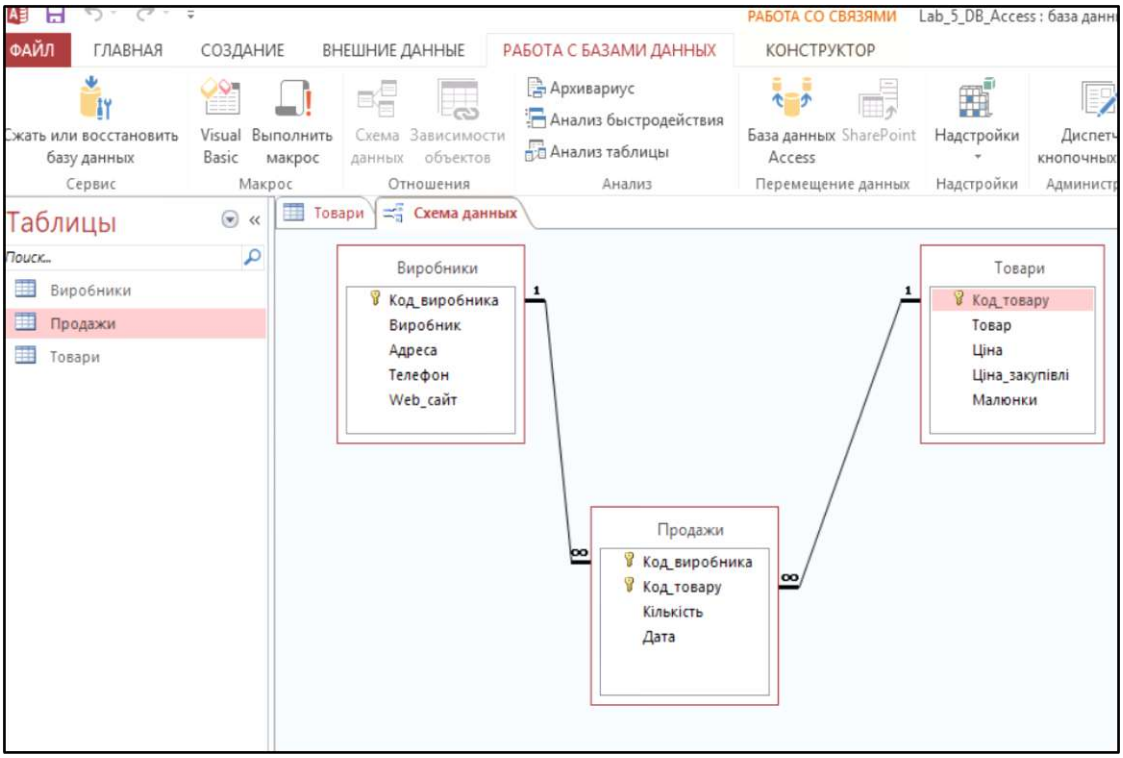


Рис. 5.7. **Схема створеної бази даних у ACCESS 2013**

5.3. Розширений рівень лабораторної роботи: створення моделі предметної області "ІС компанії з продажу товарів"

Завдання

Спроектуйте інформаційну систему для компанії, що провадять продажі товарів.

Приклад виконання

Постановка задачі: створення моделі предметної області "ІС компанії з продажу товарів"

Автоматизуються такі бізнес процеси.

1. Номенклатура продукції, що продається, складає близько 10 000 найменувань. Товари об'єднуються в ієрархічно організований каталог з необмеженою кількістю рівнів ієрархії.

2. Замовлення на купівлю товару приймають по телефону й обробляють менеджери організації. Замовлення включає необмежену кількість товарів. Покупцем може виступати фізична або юридична особа.

3. Замовлення доставляють кур'єри (у разі доставки по місту) і транспортні компанії (у разі доставки в інші регіони країни). Відвантаження товарів оформлюється витратною накладною.

4. Увесь товар ураховується на складі. Постачання товару здійснюється виробниками й оформлюється прибутковими накладними.

5. У системі має бути передбачено ведення бази даних, яка повинна зберігати інформацію про покупців, виробників, товари, замовлення, служби доставки.

5.3.1. Створення моделі (аналогічно до завдання 5.2.1)

5.3.2. Створення сутності (аналогічно до завдання 5.2.2)

Виділіть такі логічні об'єкти: Замовлення, Товар, Каталог, Покупець, Служба доставки, Виробник, Накладна (об'єкти можна назвати базовими).

5.3.3. Створення атрибутів (аналогічно до завдання 5.2.3)

Для кожної сутності визначимо перелік атрибутів і їх властивості.

Для сутності "ЗАМОВЛЕННЯ" створіть атрибути: Код_Замовлення, Дата_Замовлення, Статус_Замовлення, Спосіб_Замовлення, Спосіб_Оплати, Спосіб_Доставки, Вартість_Доставки, Покупець.

Для сутності "ПОКУПЕЦЬ" створіть атрибути: Код_Покупця, ПІБ, Телефон, Місто, Адреса.

Для сутності "ТОВАР" створіть атрибути: Код_Товар, Назва, Модель, Виробник, Ціна, Од_виміру.

Для сутності "ВИРОБНИК" створіть атрибути: Код_виробника, Виробник, Телефон, Адреса.

Для сутності "СЛУЖБА ДОСТАВКИ" створіть атрибути: Код_служби_доставки, Назва_служби.

Для сутності "НАКЛАДНА" створіть атрибути: Код_накладної, Дата_накладної, Тип_накладної.

Для сутності "КАТАЛОГ" створіть атрибути: Код_рубрики, Послідовність_ієрархі, Назва.

Для сутності "ЗАМОВЛЕННЯ_ТОВАР" створіть атрибути: Код_замовлення_товар, Кількість.

Таким чином буде створена діаграма сутностей (рис. 5.8).

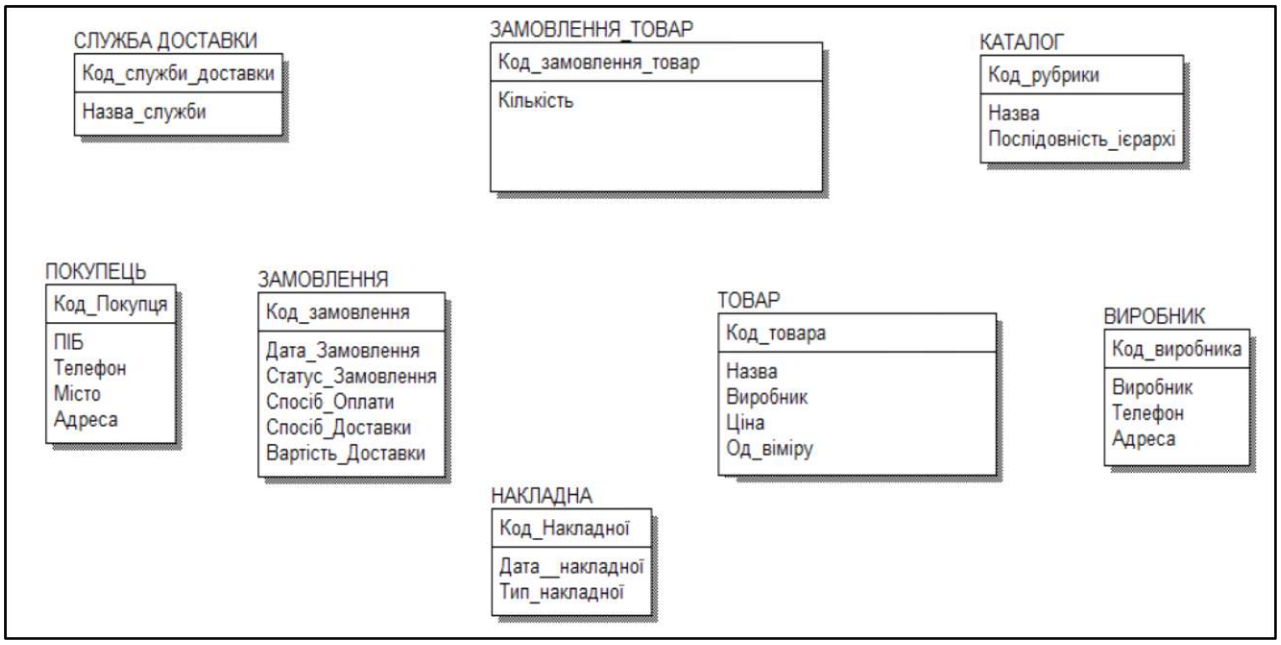


Рис. 5.8. **Діаграма сутностей без зв'язків**

5.3.4. Створення зв'язків

Завдання

Створіть:

неідентифікуючий зв'язок "Покупець робить Замовлення";

зв'язок "багато-до-багатьох" між сутностями "ЗАМОВЛЕННЯ" і "ТОВАР";

рекурсивний зв'язок для сутності "КАТАЛОГ".

Виконання

1. Клацніть на кнопці  на панелі інструментів.
2. Клацніть на батьківській сутності.
3. Клацніть на дочірній сутності.

Примітка. Якщо немає інформації про покупця, то інформація про замовлення не має сенсу. Тобто наявність атрибута Номер покупця в Замовленні обов'язкова. Така ситуація обумовлена обов'язковістю зв'язка (Властивість Nulls).

Для того щоб установити обов'язковість зв'язка, необхідно клацнути правою кнопкою миші на зв'язку та вибрати пункт **Relationship Properties** в контекстному меню.

У блоці **Nulls** вибрати позицію **No Nulls**. В цьому випадку значок на початку стрілки зникне (рис. 5.9).

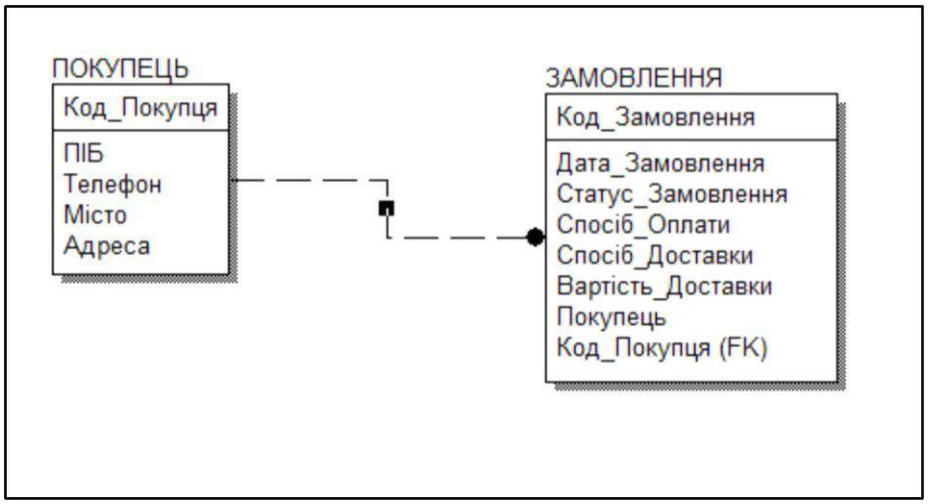


Рис. 5.9. **Відображення обов'язкового зв'язку**

Прикладом зв'язка "багато-до-багатьох" може слугувати зв'язок між сутностями "ЗАМОВЛЕННЯ" і "ТОВАР". Одне замовлення може включати багато товарів, а один товар може входити у багато замовлень.

Для редагування властивостей зв'язку слід клацнути правою кнопкою миші на зв'язку та вибрати пункт **Relationship Properties** у контекстному меню.

Наприклад, можна присвоїти зовнішньому ключу Код_покупця в сутності "ЗАМОВЛЕННЯ" функціональне ім'я Покупець (рис. 5.10).

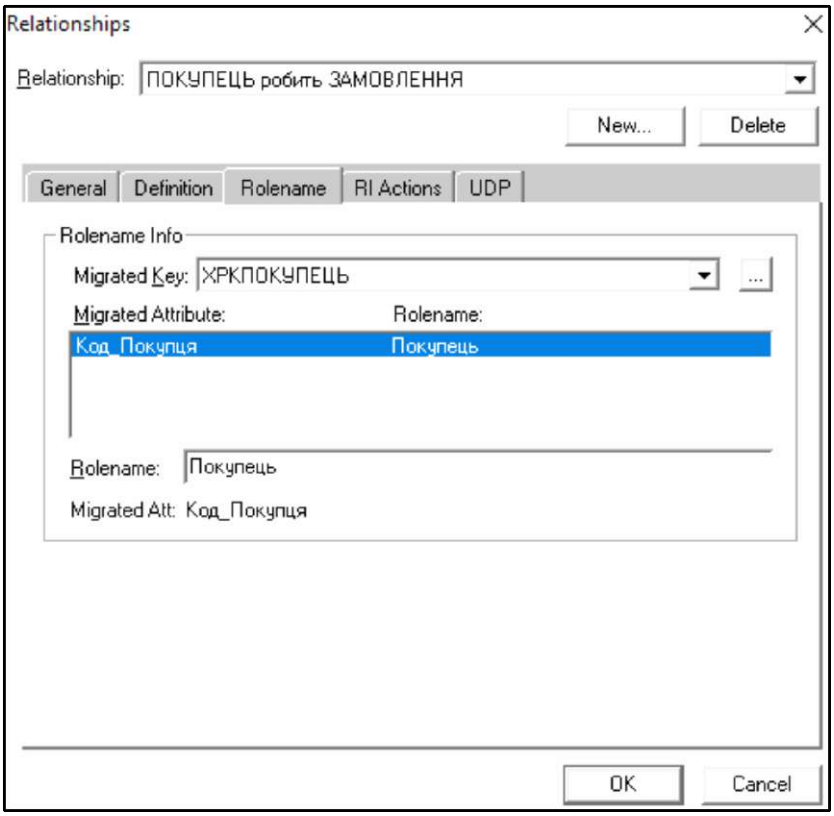


Рис. 5.10. **Вікно опису – функціональне ім'я Покупець**

За замовчуванням на діаграмі відображується тільки ім'я ролі. Щоб відобразити повне ім'я, що включає ім'я ролі та базове ім'я атрибуту, розділені точкою, необхідно в контекстному меню діаграми вибрати пункт Entity Display -> Rolename/Attribute.

Ім'я ролі необхідно вказувати обов'язково в рекурсивних зв'язках, коли та сама сутність є водночас і батьківською, і дочірньою.

Прикладом такого зв'язка може слугувати сутність "КАТАЛОГ". Тут рекурсивним зв'язком відображується входження рубрики каталогу в іншу рубрику цього ж каталогу. Зовнішній ключ є номером рубрики та відіграє тут роль підрубрики (рис. 5.11).

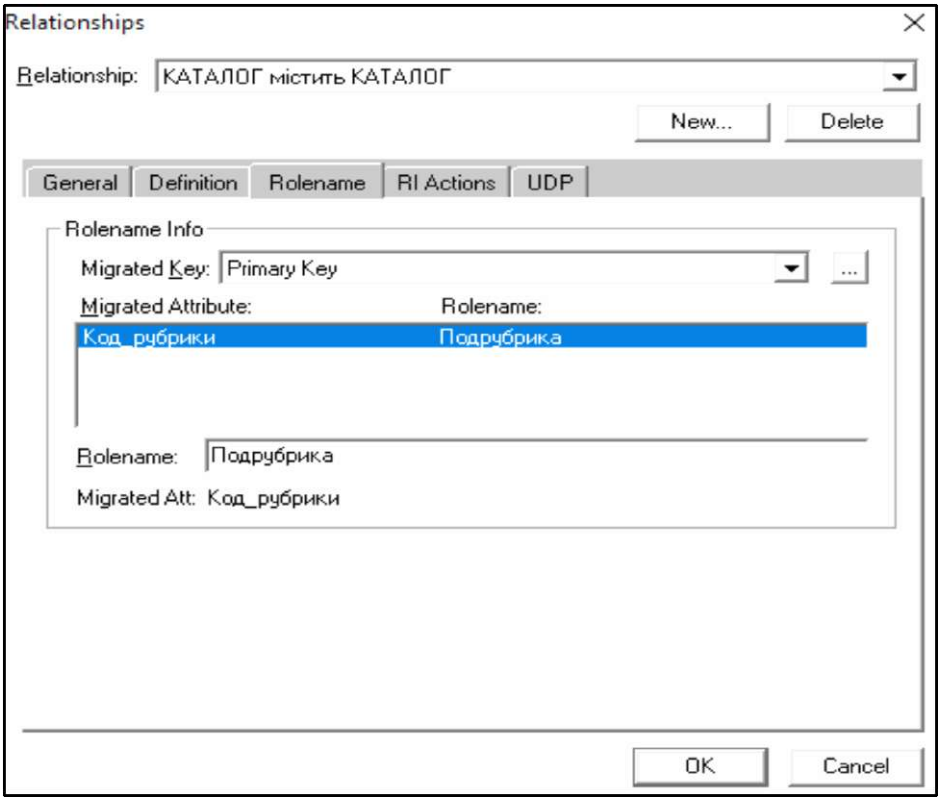


Рис. 5.11. **Створення рекурсивного зв'язка**

5.3.5. Створення ключів

Завдання

Виберіть атрибут до складу ключової сутності "ТОВАР". Створіть альтернативний ключ у сутності "ТОВАР".

Виконання

Для виконання завдання необхідно дотримуватись такої послідовності дій.

1. Виберіть необхідну сутність – "ТОВАР".

2. Наведіть курсор на атрибут, який треба внести до складу ключових. Курсор набере вигляду  .

3. Клацніть лівою кнопкою миші й, утримуючи її, перенесіть у списку атрибутів сутності вище за горизонтальну лінію.

4. Клацніть правою кнопкою миші на сутності та виберіть у контекстному меню пункт Key Group. Ви увійдете в редактор Key Groups.

5. Клацніть кнопку New і за необхідності змініть ім'я створюваного ключа. Клацніть кнопку OK.

6. Виберіть у списку Available Attributes необхідний атрибут (у цьому випадку Модель) і клацніть кнопку  . Таким чином, атрибут переміститься в список Key Group Members (рис. 5.12).

7. Клацніть кнопку OK.

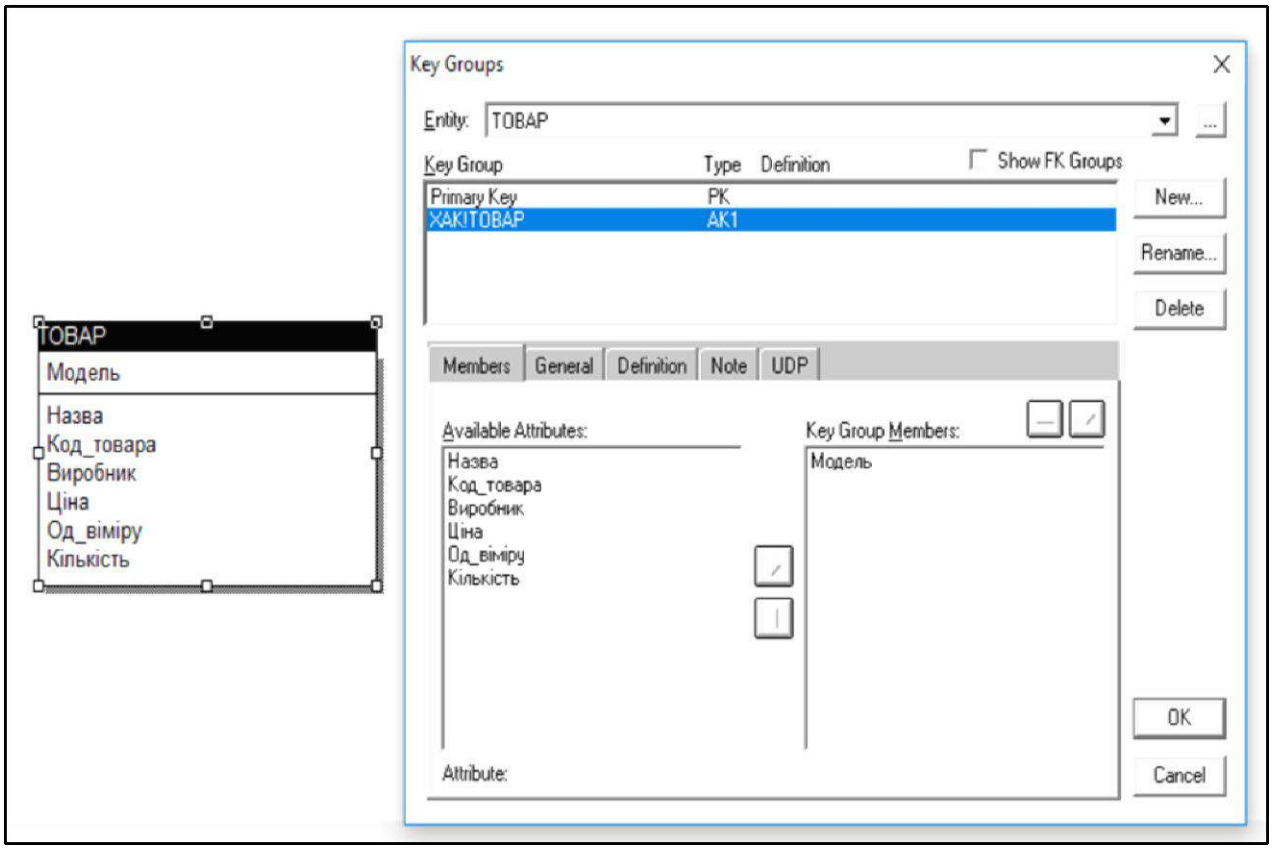


Рис. 5.12. **Приклад створення альтернативного ключа**

5.3.6. Типи сутностей та ієрархія наслідування

Завдання

Створіть категоріальний зв'язок.

Виконання

Для виконання завдання необхідно дотримуватись такої послідовності дій.

1. Створіть сутність – родовий предок ("НАКЛАДНА").

2. Створіть сутності – нащадки ("ВИТРАТНА НАКЛАДНА ТА ПРИБУТКОВА НАКЛАДНА").

3. Клацніть кнопку на панелі інструментів.

4. Клацніть спочатку на родовому предку, а потім на нащадку.

5. Для встановлення другого зв'язка в ієрархії категорії слід спочатку клацнути на символі категорії, потім на другому нащадку (рис. 5.13).

Примітка. Для кожного предка можна вказати атрибути, властиві цьому типу сутності. Наприклад, для прибуткової накладної необхідно вказати, від якого постачальника був прийнятий товар. Таким чином, діаграма набере вигляду, поданого на рис. 5.13.

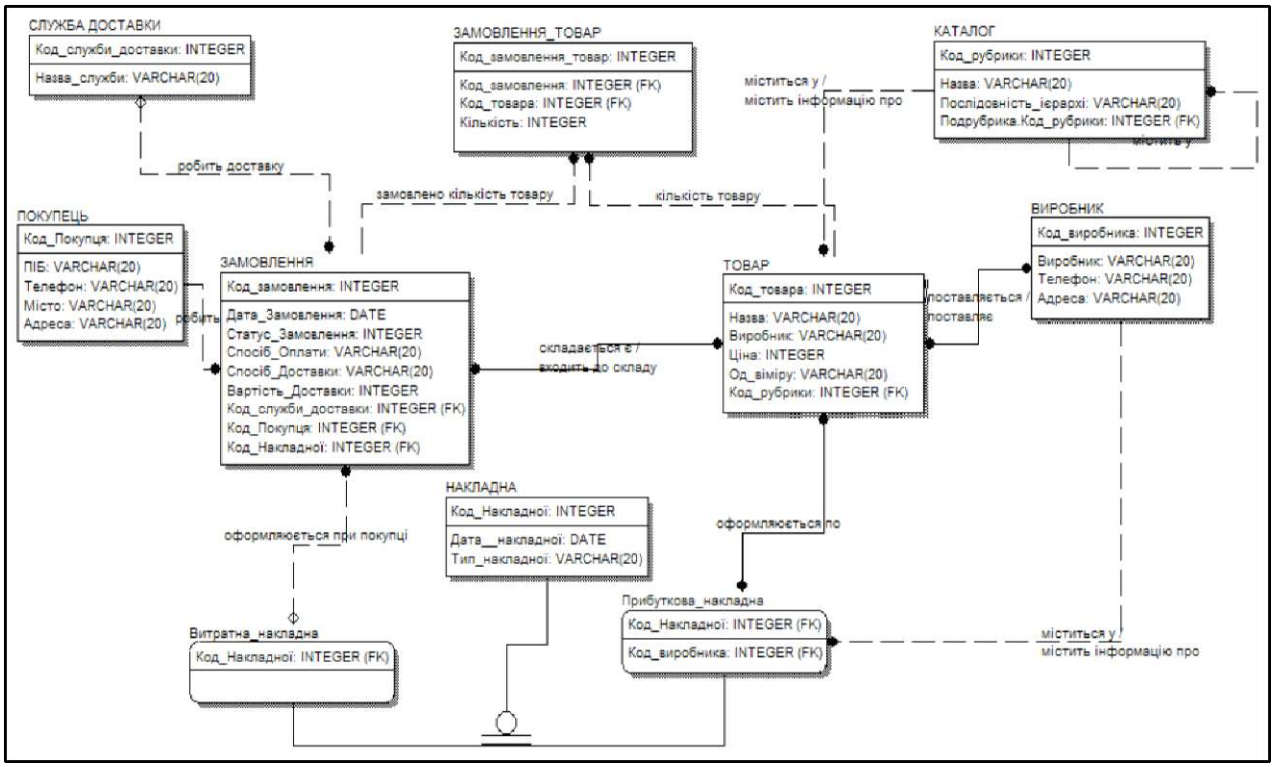


Рис. 5.13. **Вид створеної діаграми**

5.3.7. Рівні демонстрації зображення в ERwin

Завдання

Зробіть скриншот наступного рівня – демонстрація сутностей.

Виконання

Для виконання завдання необхідно вибрати пункт меню: Format -> Display -> Entity (рис. 5.14).

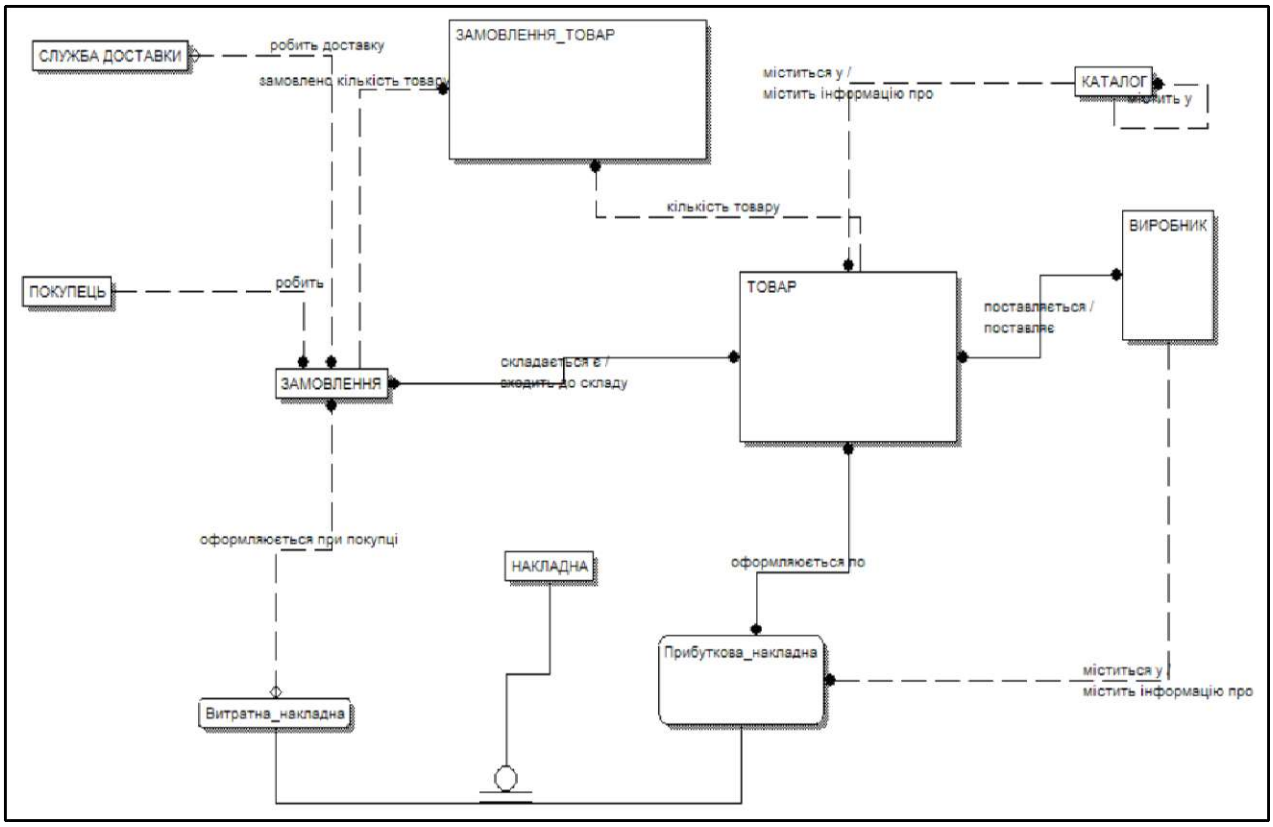


Рис. 5.14. **Рівень демонстрації визначень сутності**

5.3.8. Правила посилальної цілісності в ERwin

Завдання

Створіть правило посилальної цілісності RESTRICT для зв'язку сутностей "ПОКУПЕЦЬ" і "ЗАМОВЛЕННЯ".

Виконання

Для виконання завдання необхідно дотримуватись такої послідовності дій.

1. Зайдіть у властивості зв'язка, клацнувши правою кнопкою миші на зв'язку та вибравши пункт Relationship Properties у контекстному меню.

2. Зайдіть на вкладку RI Actions.

3. Задайте правила посилальної цілісності (рис. 5.15).

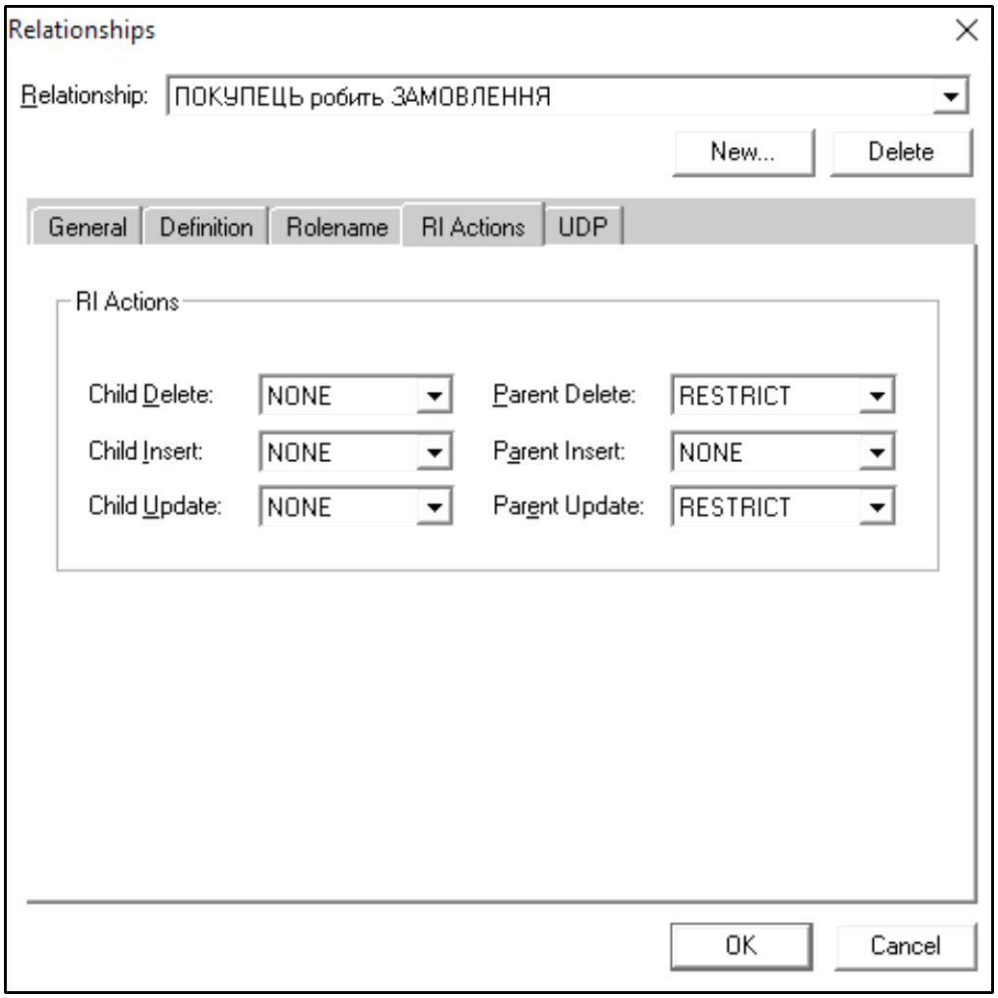


Рис. 5.15. **Вікно встановлення правил посилальної цілісності**

5.3.9. Створення фізичної моделі даних в ERwin

Завдання

Створіть подання "ЗАМОВЛЕННЯ_ПОКУПЦІ".

Виконання

Для виконання завдання необхідно дотримуватись такої послідовності дій.

1. Клацніть значок  на панелі інструментів. Курсор набере вигляду  .

2. Клацніть на будь-якому місці діаграми. На екрані з'явиться нова сутність (рис. 5.16).

3. Клацніть правою кнопкою миші на поданні та виберіть у меню пункт Database View Properties.

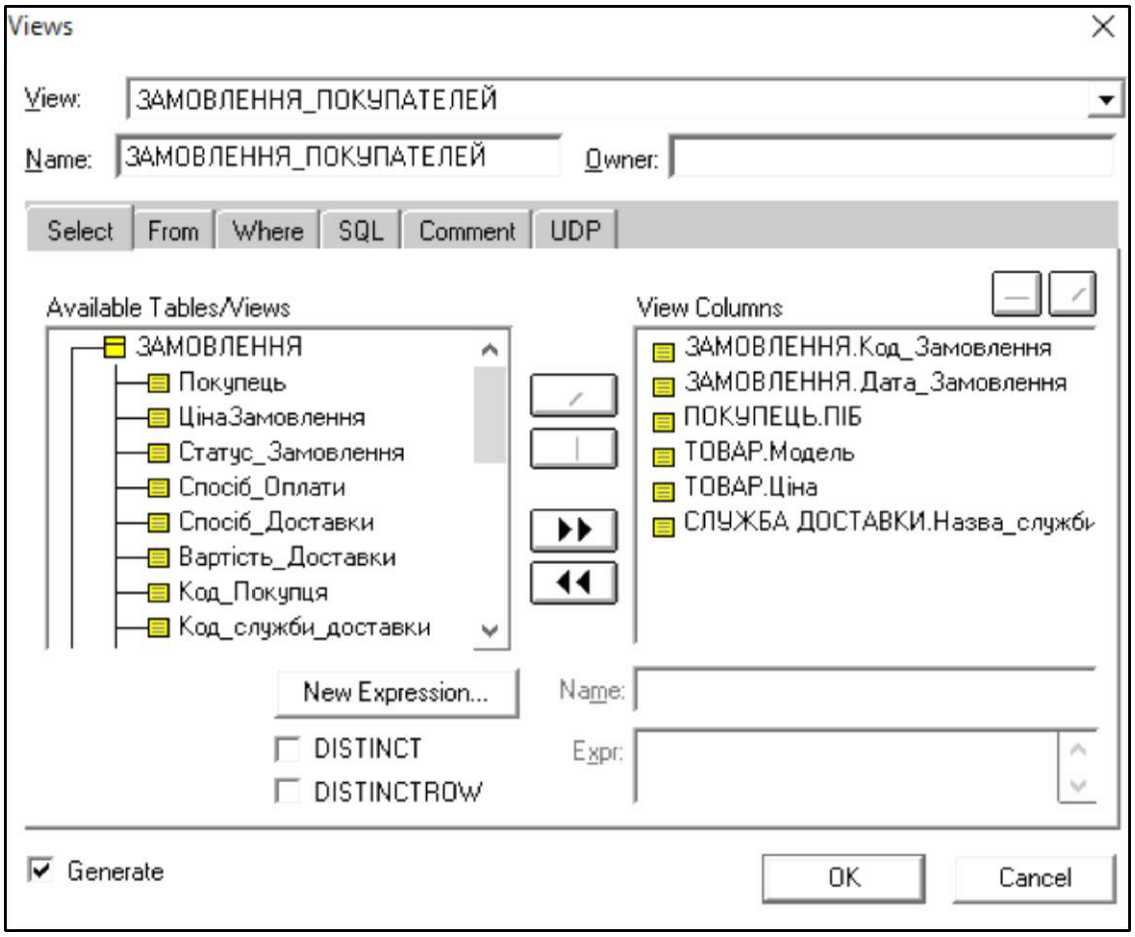


Рис. 5.16. **Вікно створення подання "ЗАМОВЛЕННЯ_ПОКУПЦІ"**

5.3.10. Позбавлення від категоріальних зв'язків в ERwin

Завдання

Позбутися від категоріального зв'язку (атрибут Тип_накладної).

Виконання

Для виконання завдання необхідно дотримуватись такої послідовності дій.

1. Виділіть створений категоріальний зв'язок для накладних і на панелі інструментів клацніть на кнопці "Supertype-Subtype Identity".

2. У вікні Supertype/Subtype Indentity Transform Wizard натисніть кнопку "Далі" і перейдіть у вікно опису перетворення категоріального зв'язку.

3. Введіть ім'я перетворення і його визначення. Клацніть кнопку "Далі". Потрапляємо у вікно "Summary".

4. Клацніть кнопку "Готово".

У результаті отримуємо модель, показану рис. 5.17.

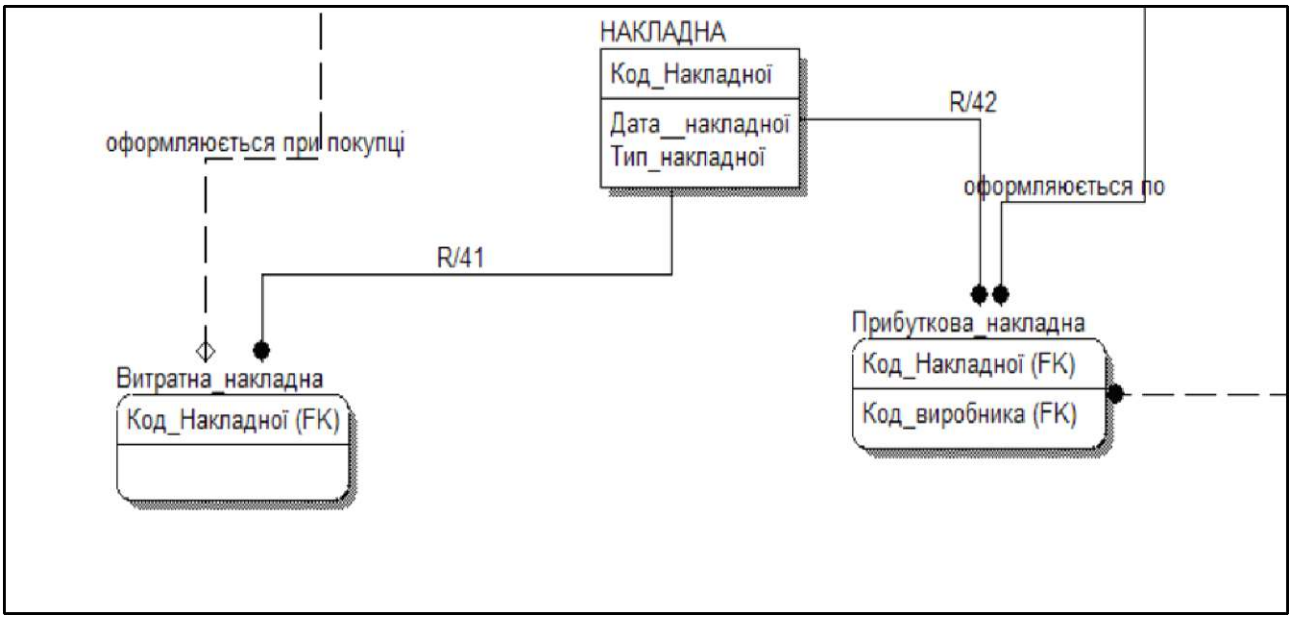


Рис. 5.17. **Фрагмент моделі з перетвореним категоріальним зв'язком**

Надалі, позбавившись від усіх зв'язків "багато-до-багатьох", можна приступати до формування схеми бази даних у рамках конкретної СКБД.

5.3.11. Зворотне проектування в ERwin

Завдання

Виконайте зворотне проектування.

Виконання

Для виконання завдання необхідно дотримуватись такої послідовності дій.

1. Виберіть пункт меню Tools -> Reverse Engineer. Після цього виникає діалог Select Template, в якому треба вибрати шаблон діаграми, потім – діалог вибору СКБД.

2. Задайте опції зворотного проектування.

3. Виберіть базу даних і виконайте підключення (Connect).

У результаті отримана згенерована модель (рис. 5.18).

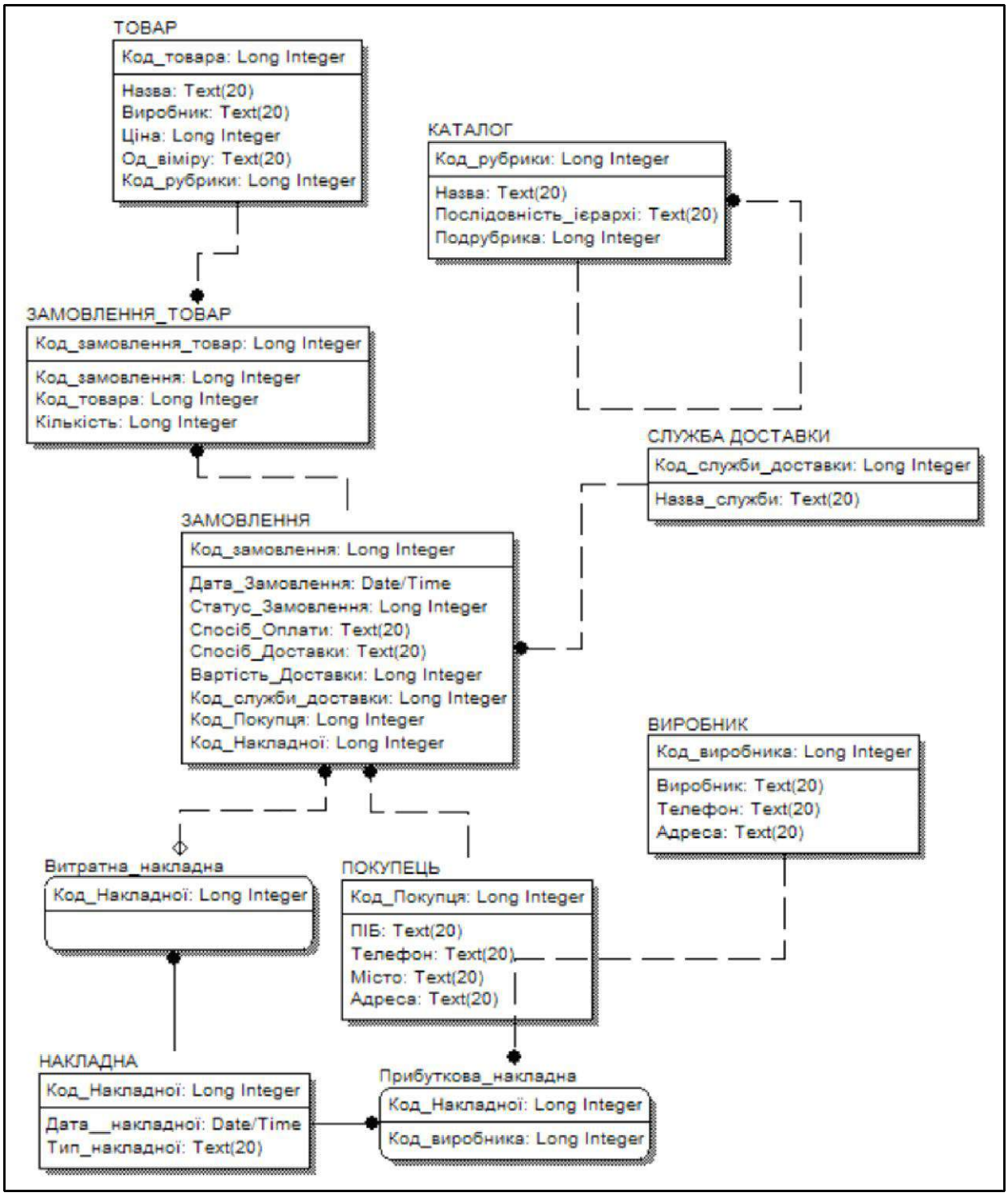


Рис. 5.18. **Згенерована модель за зворотного проектування**

Завдання для самостійного виконання

1. Зробіть скриншоти таких рівнів: демонстрація атрибутів сутностей, демонстрація первинних ключів, демонстрація визначень сутностей, демонстрація піктограм.

2. Створіть область для ілюстрації бізнес-функцій відділу продаж, яка включатиме сутності, що належать до оформлення замовлення: Покупець, Замовлення, Товар, Служба доставки.

3. Створіть новий домен ЦІНА та новий атрибут ЦінаЗамовлення в моделі, використовуючи опис доменів.

4. Створіть правило валідації для колонки Тип таблиці "СЛУЖБА ДОСТАВКИ". У описі предметної області задано, що замовлення можуть доставлятися або кур'єрами, або транспортними компаніями.

5. Виконайте пряме проектування логічної моделі даних.

6. Створіть звіт "СУТНІСТЬ" щодо лабораторної роботи за допомогою інструменту – Data Browser, в якому подайте логічну та фізичну схеми БД з CASE-інструменту, а також схему БД, отриману в Access:

6.1. Відформатуйте звіт: змініть сортування даних, черговість колонок, зробіть колонку невидимою, задайте її стиль.

6.2. Виконайте експорт набору даних "СУТНІСТЬ" у форматі DDE.

6.3. Перевірте помилки у створеній моделі.

6.4. Створіть подання звіту.

7. Створіть звіт з лабораторної роботи за допомогою Report Template Builder.

8. Побудуйте інформаційну модель для індивідуально обраної предметної області (див. табл. 4.36) – це базовий рівень лабораторної роботи.

9. Побудуйте інформаційну модель для індивідуально обраної предметної області (див. табл. 4.36) – це розширений рівень лабораторної роботи.

Лабораторна робота 6
Аналітична обробка даних

Цілі роботи

1. Набуття практичних навичок роботи зі сховищами даних у середовищі *Visual Studio* або *Management Studio*.

2. Набуття практичних навичок із створення зведених таблиць і діаграм у середовищі *Excel* на основі даних сховища SQL Server.

3. Набуття практичних навичок з проектування, створення та заповнення сховища даних засобами мови SQL.

Перед виконанням роботи студент повинен знати:

основи використання *Visual Studio* для роботи з базами даних;
основні об'єкти бази даних MS SQL Server;
основні команди мови SQL;
основи побудови зведених таблиць і діаграм.

Після виконання лабораторної роботи студент повинен уміти:

самостійно проектувати та створювати сховища даних;
заповнювати оперативні бази та сховища даних значною кількістю даних;
виконувати аналіз даних у середовищі *Excel* на основі даних сховища.

Хід роботи

6.1. Порівняльний аналіз оперативної бази даних і сховища даних.
6.2. Аналіз даних у середовищі Excel на основі даних сховища.
6.3. Проектування сховища даних.
6.4. Створення і заповнення сховища даних.
6.5. Створення зведених таблиць і діаграм у середовищі Excel на основі даних сховища.

Форма звітності

За результатами виконання роботи необхідно оформити електронний звіт засобами Word.

В електронному звіті щодо п. 6.1 слід помістити список схем бази даних та їхнє призначення, а також пояснити призначення таблиць ***SalesOrderHeader і SalesOrderDetail***, що містяться у схемі ***Sales.*** Занесіть ваші міркування щодо того, як утворені ці таблиці і як вони можуть бути пов'язані між собою.

У звіті щодо п. 6.2 подайте скриншоти (не менше трьох) з результатами експериментів. Порівняйте динаміку продажів товарів червоного та інших кольорів.

У звіті щодо п. п. 6.3, 6.4 подайте тексти скриптів для створення і заповнення таблиць вимірювань і фактів, а також скриншоти даних цих таблиць.

У звіті щодо п. 6.5 подайте скриншот панелі моніторингу.

Для кожної діаграми надайте економічне трактування процесів, що зображені на ній.

Критерії оцінювання

У 12-бальній системі оцінка результату захисту лабораторної роботи формується за такими правилами.

1. За кожне завдання з п.п. 6.1 і 6.2 може бути виставлено від 0 до 2 балів.

2. За завдання 6.3 може бути виставлено від 0 до 1 бала.

3. За завдання 6.4 може бути виставлено від 0 до 3 балів.

4. За завдання 6.5 може бути виставлено від 0 до 1 бала.

5. За кожне завдання із п. "Завдання для самостійного виконання" може бути виставлено від 0 до 3 балів.

6. За кілька варіантів розв'язання одного із завдань додається 1 бал.

7. За вибір варіанту, який з кількох варіантів розв'язання є оптимальним, та обґрунтування вибору додається 1 бал.

8. За запізнення із захистом лабораторної роботи знімається 2 бали за кожний тиждень.

Отримана кількість балів за відповіді на кожне завдання лабораторної роботи підсумовується. У результаті такого підрахунку студентом може бути отримано від 0 до 12 балів.

Рекомендована література: [3; 18; 21].

Основні поняття

OLTP (від англ. On-Line Transaction Processing) – це системи, що орієнтовані на операційну (транзакційну) обробку даних. Вони виконують таку функцію: одночасне виконання великої кількості коротких транзакцій від великого числа користувачів.

Ці системи мають такі *характеристики*:

підтримка великої кількості користувачів;

малий час відгуку на запит;

відносно короткі запити;

участь у запитах невеликої кількості таблиць.

Системи підтримки прийняття рішень (СППР), або Decision Support Systems (DSS) – це системи, що орієнтовані на аналітичну обробку даних.

Вони виконують такі *функції*:

аналіз даних;

моделювання процесів предметної області;

прогнозування;

знаходження залежностей між даними;

проведення аналізу "що якщо".

Ці системи мають такі *характеристики:*

використання великих обсягів даних;

додавання в систему нових даних відбувається відносно рідко, великими блоками;

дані, що додані в систему, зазвичай ніколи не видаляються;

перед завантаженням дані проходять різні процедури "очищення";

невелика кількість користувачів (аналітики).

В основу аналітичних систем закладена концепція сховища даних. Ця концепція визначає процес збирання, відсіювання, попередньої обробки та накопичення даних з метою:

довготривалого зберігання даних;

надання результівної інформації користувачам у зручній формі для статистичного аналізу та створення аналітичних звітів.

Концепція інтелектуального аналізу даних визначає завдання пошуку функціональних і логічних закономірностей в накопиченій інформації, побудову моделей і правил, які пояснюють знайдені аномалії і/або прогнозують розвиток деяких процесів.

Концепція OLAP – це комплексна інтерактивна обробка даних, яка використовує методи багатовимірного аналізу даних з метою підтримки процесів прийняття рішень. Теоретично засоби OLAP можна застосовувати безпосередньо до оперативних даних або їх точних копій (щоб не заважати оперативним користувачам).

Сховище може зберігати такі *категорії даних*:

детальні дані відповідають елементарним подіям, що фіксуються OLTP-системами (продажі, експерименти тощо). Їх розподіляють на вимірювання (дані, необхідні для опису подій: міста, події, люди та ін.) і факти (дані, що відображають сутність події: кількість проданого товару, результати експериментів тощо);

агреговані дані, у свою чергу, розподіляють на такі види:

адитивні – це числові фактичні дані, які можна підсумувати за всіма вимірами;

напівадитивні – це числові фактичні дані, які можна підсумувати тільки за певними вимірами;

метадані – відомості про дані, що зберігаються у сховищі даних. Метадані повинні відповідати на такі запитання:

що – опис об'єктів предметної області, які зберігаються у сховищі даних;

хто – опис категорій користувачів, які використовують дані, та їхні права доступу;

де – опис місця розташування серверів, робочих станцій, місць зберігання даних і розподіл даних між ними;

коли – опис часу виконання різних операцій над даними;

чому – опис причин, що призвели до виконання над даними тих чи інших операцій.

ETL (від англ. Extract, Transform, Load – витяг, перетворення, завантаження) – один з базових процесів управління сховищами даних, а також найменування класу утиліт автоматизації цього процесу.

Програми ETL використовують для перенесення інформації із застосувань у нові або для передавання операційних даних у системи бізнес-інтелекту такі, як сховища або кіоски даних. ETL включає вибірку даних із зовнішніх джерел, їх перетворення відповідно до вимог бізнес-моделі, завантаження перетворених даних у цільову систему (наприклад, сховище даних).

Кожен з етапів ETL у реальності досить складний.

В якості зовнішніх джерел інформації можуть виступати різні інформаційні системи, формати зберігання даних яких і процедури їх отримання можуть істотно різнитися.

ETL-процес не зводиться виключно до технічного перетворення форматів. Дані з різнорідних джерел повинні бути уніфіковані з точки зору бізнес-правил, єдності застосовуваних систем кодування інформації, класифікаторів і довідників. Процес повинен враховувати і особливості бізнес-процесів компанії, в тому числі, функціонування джерел даних окремих інформаційних систем, частоту оновлення даних у них тощо.

OLAP-технологія надає для аналізу дані у вигляді багатовимірних (і отже, нереляційних) наборів даних, які називають *багатовимірними кубами* (гіперкуб, метакуб, куб фактів), осі яких містять параметри, а клітинки – залежні від них агрегатні дані. Гіперкуб є концептуальною логічною моделлю організації даних, а не фізичною реалізацією їх зберігання, оскільки зберігатися такі дані можуть і в реляційних таблицях.

Над гіперкубом можуть виконуватися такі **операції**:

зріз (slice) – формується підмножина багатовимірного масиву даних, яка відповідає єдиному значенню одного або декількох елементів вимірювань, що не входять у цю підмножину;

обертання (rotate) – зміна розташування вимірювань, що подані у звіті чи на сторінці, що відображається. Дає користувачам можливість побачити дані, які згруповані за іншими вимірами;

консолідація (roll up) і *деталізація* (drill down) – операції, які визначають перехід вгору в напрямі від детального (down) подання даних до агрегованого (up), і навпаки. Це взаємодоповнювальні операції, які використовують ієрархію вимірів і параметри для агрегування.

Системи оперативної аналітичної обробки реляційних даних (ROLAP) дозволяють подавати дані, що зберігаються в реляційній базі, в багатовимірної формі, забезпечуючи перетворення інформації в багатовимірну модель через проміжний шар метаданих. У цьому випадку гіперкуб емулюється СКБД на логічному рівні.

Для більшості сховищ даних найбільш ефективним способом моделювання N-вимірного куба фактів є схема "зірка" (star schema). Схема сніжинки утворюється зі схеми зірка в разі нормалізації таблиць вимірів останньої.

Основними складовими структури сховищ даних є таблиці вимірів (dimension tables) і таблиця фактів (fact table).

Таблиці вимірів містять незмінювані або рідко змінювані дані. У кожній таблиці вимірів перераховані можливі значення одного з вимірів гіперкуба. У переважній більшості випадків ці дані надають по одному запису для кожного члена нижнього рівня ієрархії у вимірі. Таблиці вимірів також містять як мінімум одне описове поле (зазвичай з ім'ям члена виміру) і, як правило, цілочисельне ключове поле (зазвичай це сурогатний ключ) для однозначної ідентифікації члена виміру. Кожна таблиця виміру повинна знаходитися у відношенні "один-до-багатьох" з таблицею фактів.

Причина, за якою дана схема названа "зіркою", така: кінці зірки утворюються таблицями вимірів, а їхні зв'язки з таблицею фактів, розташованої в центрі, утворюють промені.

Таблиця фактів є основною таблицею сховища даних. Як правило, вона містить відомості про об'єкти або події, сукупність яких буде надалі аналізуватися. Якщо проводити аналогію з багатовимірною моделлю, то рядок таблиці фактів відповідає клітинці гіперкуба. Зазвичай говорять про чотири найпоширеніші *типи фактів*. До них відносять такі:

факти, що пов'язані з транзакціями (transaction facts). Вони засновані на окремих подіях (типовими прикладами яких є телефонний дзвінок або зняття грошей з рахунку у банкоматі);

факти, що пов'язані з "миттєвими знімками" (snapshot facts). Засновані на стані об'єкта (наприклад, банківського рахунку) в певні моменти часу, (наприклад, на кінець дня чи місяця). Типовими прикладами таких фактів є обсяг продажів за день або денна виручка;

факти, що пов'язані з елементами документа (line-item facts). Засновані на тому чи іншому документі (наприклад, рахунку за товар або послугу) та містять детальну інформацію про елементи цього документа (наприклад, кількість, ціну, відсоток знижки);

факти, що пов'язані з подіями або станом об'єкта (event or state facts). Подають виникнення події без подробиць про неї (наприклад, просто факт продажу або факт відсутності такої без інших подробиць).

Отже, основними відмінностями сховищ даних від операційних баз даних є таке:

містять історичні дані;

зберігають докладні відомості, а також частково і значно узагальнені дані;

дані в основному є статичними;

нерегламентований, неструктурований і евристичний спосіб обробки даних;

середня і низька інтенсивність обробки транзакцій;

непередбачуваний спосіб використання даних;

призначені для проведення аналізу;

орієнтовані на предметні області;

підтримують прийняття стратегічних рішень;

обслуговують відносно малу кількість працівників виконавчої ланки.

Практична частина

6.1. Порівняльний аналіз оперативної бази даних і сховища даних

Попередні зауваження

У навчальних цілях компанією Microsoft розроблено фіктивну компанію Adventure Works Cycles.

Компанія Adventure Works Cycles – це велика транснаціональна виробнича компанія, яка випускає і реалізує металеві та композитні велосипеди на ринках Північної Америки, Європи та Азії. Штаб-квартира компанії Adventure Works Cycles знаходиться в місті Боселлі, штат Вашингтон. У ній працюють 500 співробітників. Крім того, компанія Adventure Works Cycles має в своєму складі кілька груп збуту на регіональних ринках.

На основі вигаданих даних про роботу цієї компанії у середовищі *SQL Server* створено такі приклади баз даних:

оперативна база даних AdventureWorks;

сховище даних AdventureWorksDW;

служба аналітики AdventureWorksAS.

У цій лабораторній роботі використовуються перші дві бази даних.

Залежно від версії *Visual Studio* використовують різні зразки баз даних. Для *Visual Studio 2010* прийнятними є такі: AdventureWorks2008_Data.mdf та AdventureWorksDW_Data.mdf. Друга з них створена у середовищі попередньої версії *SQL Server 2005*. В інструкції описано, як змінити рівень цієї бази даних.

Для *Visual Studio* вищих версій рекомендуються такі зразки баз даних: AdventureWorks2012_Data.mdf та AdventureWorksDW2012_Data.mdf. Дані цих баз відрізняються від попередніх тільки датами. У них номери років збільшено на чотири.

Завдання 1

Ознайомтесь зі структурою оперативної бази даних AdventureWorks.

Виконання

Для виконання завдання необхідно дотримуватись такої послідовності дій.

1. Відкрийте Visual Studio.

2. Створіть проект Windows Forms мовою C# з ім'ям ***appData***.

3. Додайте до проекту оперативну базу даних AdventureWorks:

3.1. клацніть правою клавішею миші на позначці проекту у вікні **Solution Explorer** і з контекстового меню виберіть команду **Add – Existing Item**;

3.2. виберіть файл ***AdventureWorks2008_Data.mdf***, якщо лабораторна робота виконується у *Visual Studio 2010*. Для більш високих версій *Visual Studio* виберіть файл ***AdventureWorks2012_Data.mdf***.

4. Дочекайтеся, поки відкриється вікно Server Explorer з підключеною оперативною базою даних; розкрийте її вузол, перегляньте список таблиць бази даних і дані деяких таблиць.

У звіті з лабораторної роботи помістіть відповідні скриншоти з підрисунковими підписами.

Примітка. Якщо під час розкриття вузла бази даних у старших версіях *Visual Studio* з'являється повідомлення про неможливість під'єднання бази даних до сервера, клацніть правою клавішею миші на позначці бази даних, з контекстового меню виберіть команду **Modify Connection** і у вікні, що з'явилося, клацніть кнопку **OK**.

5. Щоб отримати більш повне уявлення про структуру оперативної бази даних, відкрийте файл ***AdvWorksOLTPSchemaVisio.vsd*** у застосуванні Visio.

У звіті з лабораторної роботи помістіть список схем бази даних та їхнє призначення, а також поясніть призначення таблиць ***SalesOrderHeader*** і ***SalesOrderDetail***, що містяться у схемі ***Sales***.

Завдання 2

Ознайомтесь зі структурою сховища даних AdventureWorksDW.

Виконання

Для виконання завдання необхідно дотримуватись такої послідовності дій.

1. Додайте до проекту базу даних AdventureWorksDW. Для цього:

1.1. клацніть правою клавішею миші на позначці проекту у вікні **Solution Explorer** і з контекстового меню виберіть команду **Add – Existing Item**;

1.2. виберіть файл ***AdventureWorksDW_Data.mdf***, якщо лабораторна робота виконується в *Visual Studio 2010*. Для більш високих версій *Visual Studio* виберіть файл ***AdventureWorksDW2012_Data.mdf***.

2. Дочекайтеся, поки відкриється вікно Server Explorer з підключеною оперативною базою даних; розкрийте її вузол, перегляньте список таблиць бази даних і дані деяких таблиць.

У звіті з лабораторної роботи помістіть відповідні скриншоти з підрисунковими підписами, а також укажіть, у чому полягає основна відмінність імен сховища даних від оперативної бази даних.

3. Перегляньте дані таблиць ***FactInternetSales*** і ***DimDate*** (у *Visual Studio 2010* ***DimTime***).

У звіті з лабораторної роботи помістіть ваші міркування щодо того, як утворені таблиці ***FactInternetSales*** і ***DimDate*** і як вони можуть бути пов'язані між собою.

6.2. Аналіз даних у середовищі Excel на основі даних сховища

Постановка задачі візуалізації даних

Побудуйте панель моніторингу продажів товарів через Інтернет. На панелі відобразіть вартість проданих товарів у різних країнах (у тому числі і у різних містах) за роками та місяцями протягом усього періоду роботи компанії Adventure Works Cycles. Надайте можливість прослідкувати дані щодо товарів кожного виду, а також за їхнім кольором. Панель повинна бути інтерактивною, щоб зміни в одному елементі відображалися в інших.

Зовнішній вигляд панелі моніторингу продажів зображено на рис. 6.1.

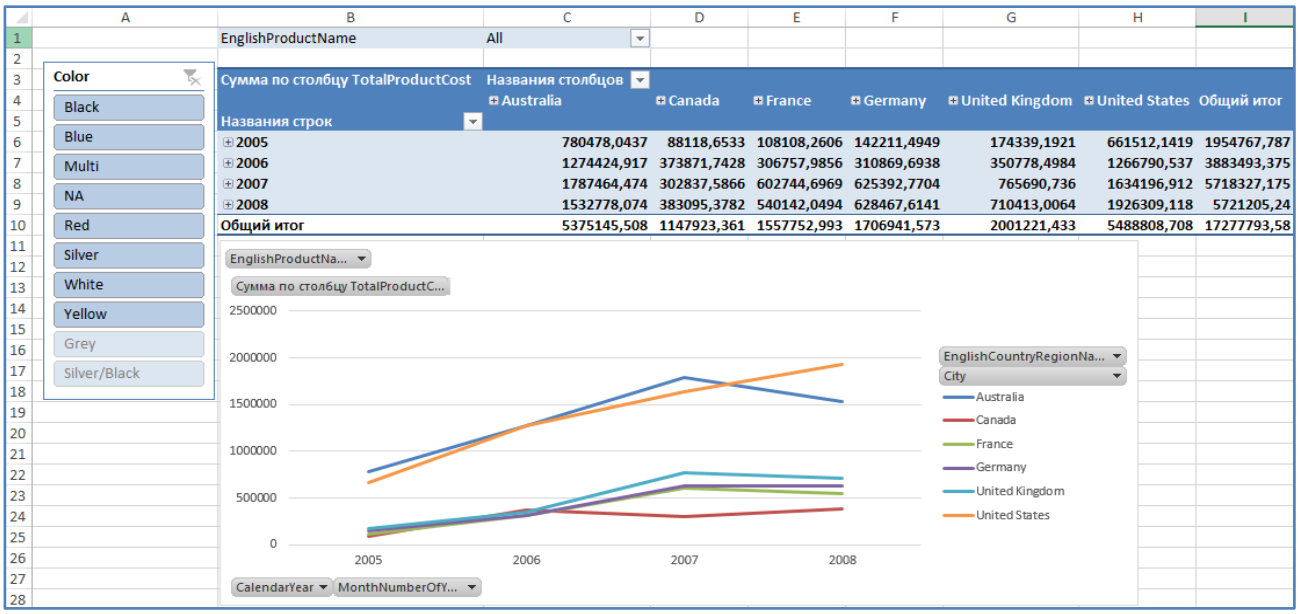


Рис. 6.1. **Панель моніторингу продажів**

Для розв'язання задачі будуть потрібні дані сховища AdventureWorksDW, що подано у табл. 6.1.

Таблиця 6.1

Дані для побудови панелі моніторингу продажів

Таблиця	Поле
1	2
DimCustomer	–
DimDate	CalendarYear
	MonthNumberOfYear
DimGeography	City
	EnglishCountryRegionName

1	2
DimProduct	EnglishProductName
	Color
FactInternetSales	TotalProductCost

Примітка: таблиця DimCustomer потрібна для зв'язування таблиць FactInternetSales та DimGeography.

Завдання 1

Створіть файл під'єднання до сховища даних в Excel.

Виконання

Для виконання завдання необхідно дотримуватись такої послідовності дій.

1. Створіть нову книгу *Excel*, назвавши її ***Аналіз***.
2. Двічі клацніть на ярличку аркуша книги *Excel* і перейменуйте його на ***Моніторинг продажів***.
3. Клацніть на клітинці **A1** і перейдіть у вкладку **Данные**.
4. Клацніть кнопку **Получение внешних данных** і зі списку, що відкрився, виберіть елемент **Из других источников**, а потім – **Из мастера подключения данных** (рис. 6.2).

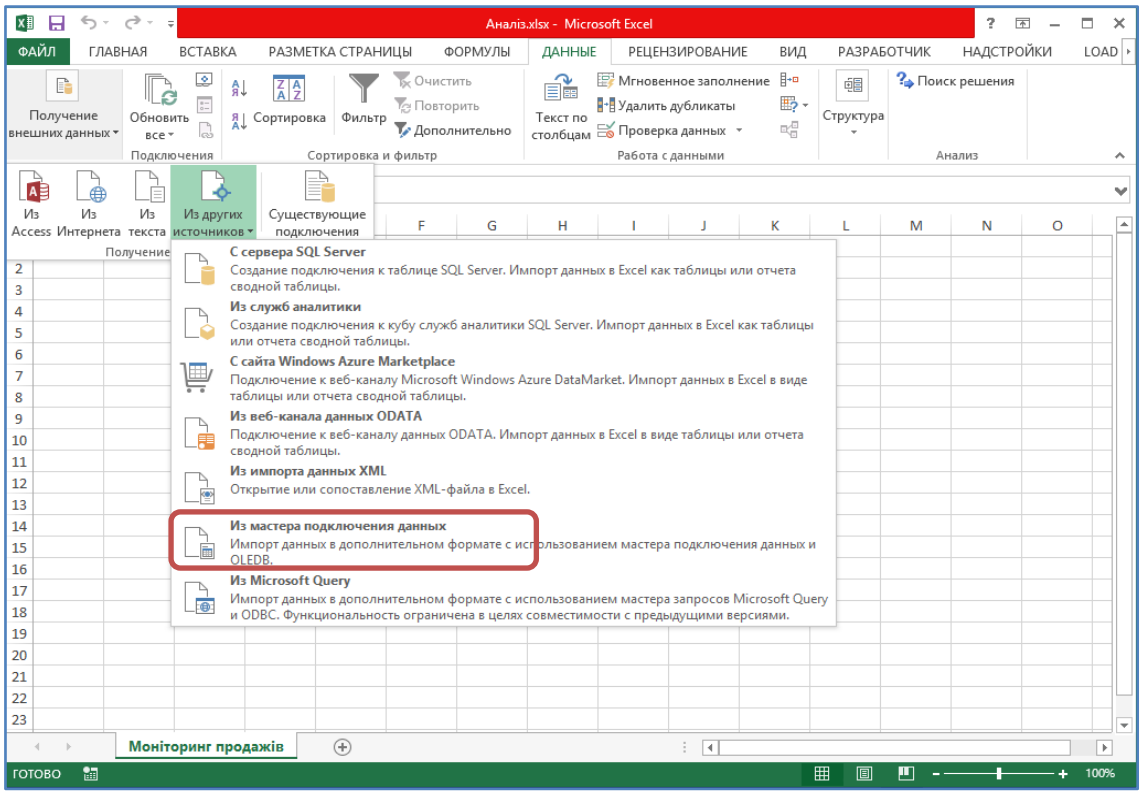


Рис. 6.2. **Вибір команди для отримання даних**

5. Виберіть елемент **Дополнительно** у першому вікні Майстра під'єднання до даних і клацніть кнопку **Далее** (рис. 6.3).

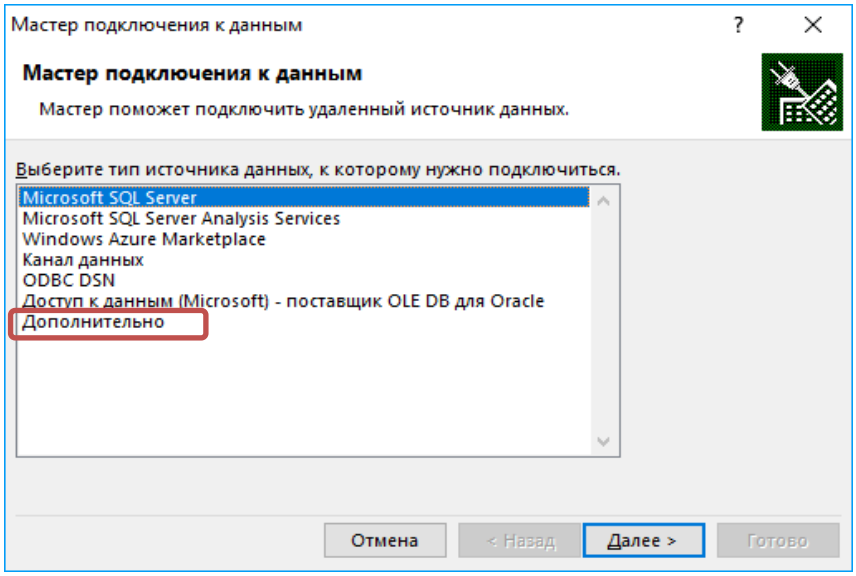


Рис. 6.3. **Вибір елемента Дополнительно**

6. Виконайте такі дії у двох вкладках вікна **Свойства канала передачи данных**:

6.1. виберіть елемент **SQL Server Native Client 10.0** у вкладці **Поставщик данных** і не клацайте кнопку **Далее** (рис. 6.4);

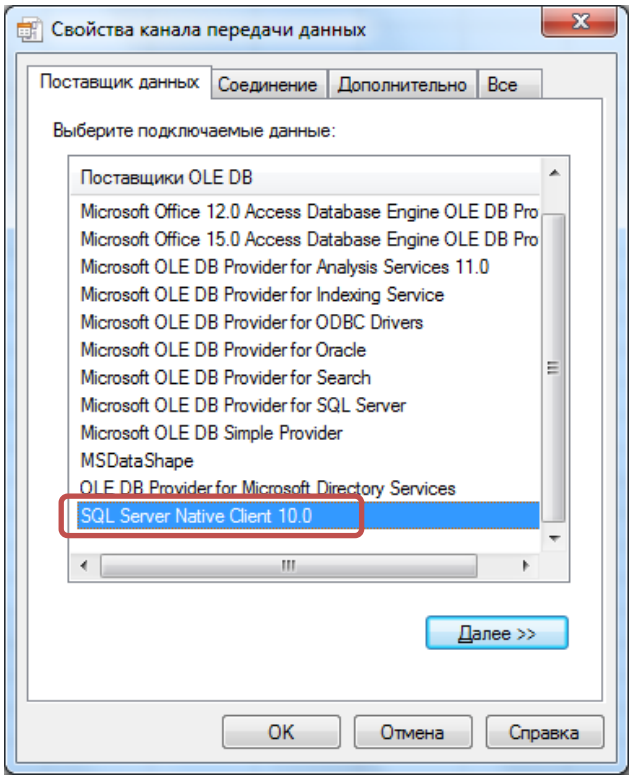


Рис. 6.4. **Вибір елемента SQL Server Native Client**

Примітка: у п. 6.1 указано елемент **SQL Server Native Client 10.0**, що притаманний *Visual Studio 2010*. Для старших версій *Visual Studio* він має більший номер.

6.2. перейдіть у вкладку **Соединение** та введіть у поле зі списком **Select or enter a server name** ім'я сервера. Для *Visual Studio 2010* треба ввести ***.\SQLExpress***; для *Visual Studio 2012, 2013* – ***(localDB)\v11.0***; а для вищих версій – ***(localDB)\MSSQLLocalDB***.

Потім виберіть перемикач **Use Windows NT Integrated security**, який означає, що до сховища даних будуть застосовуватися ті самі логін і пароль, які були введені під час входу в операційну систему Windows.

Після цього виберіть перемикач **Attach a database file as a database name** і введіть у текстове поле **Using the filename** повний шлях до файла сховища даних у проекті *Visual Studio*, використавши кнопку ... .

На завершення роботи у цьому вікні перевірте з'єднання з базою даних, клацнувши кнопку **Test Connection**; клацніть кнопку **ОК**.

На рис. 6.5 показано вкладку **Соединение** зі встановленими параметрами для *Visual Studio 2013*.

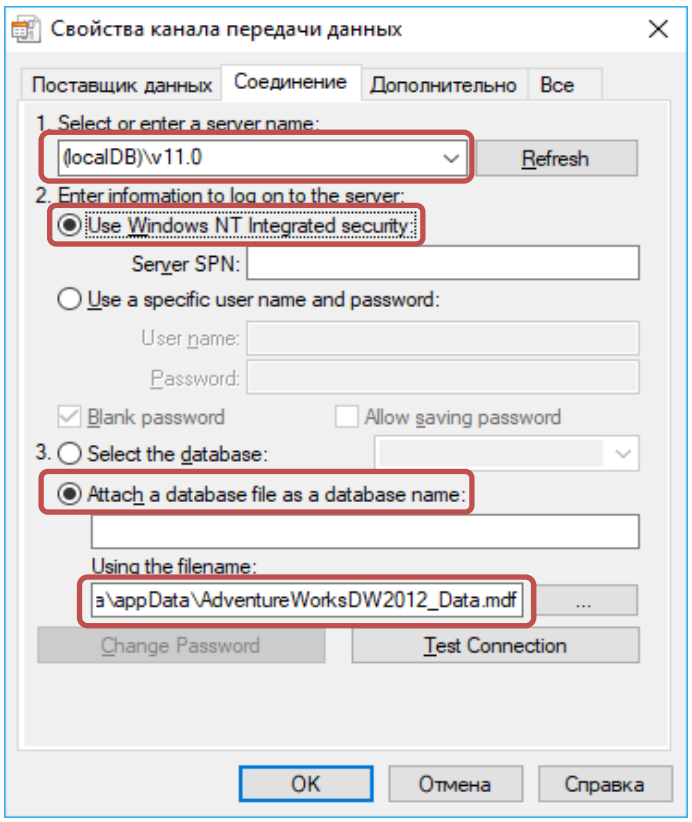


Рис. 6.5. **Вкладка Соединение із встановленими параметрами**

7. Вимкніть прапорець у вікні **Выбор базы данных и таблицы** та клацніть кнопку **Далее** (рис. 6.6).

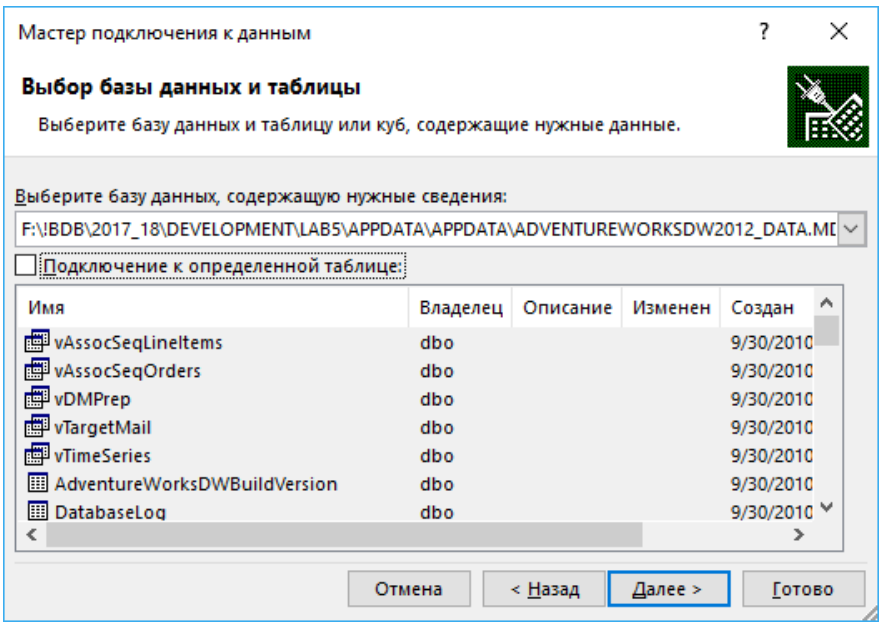


Рис. 6.6. **Вікно Выбор базы данных и таблицы**

8. Введіть зрозуміле ім'я файла під'єднання до джерела даних та його опис у вікні **Сохраните файл подключения данных и завершите работу** (рис. 6.7). За цим ім'ям можна буде під'єднуватися до сховища даних у наступних сеансах роботи з *Excel*. Потім клацніть кнопку **Готово**.

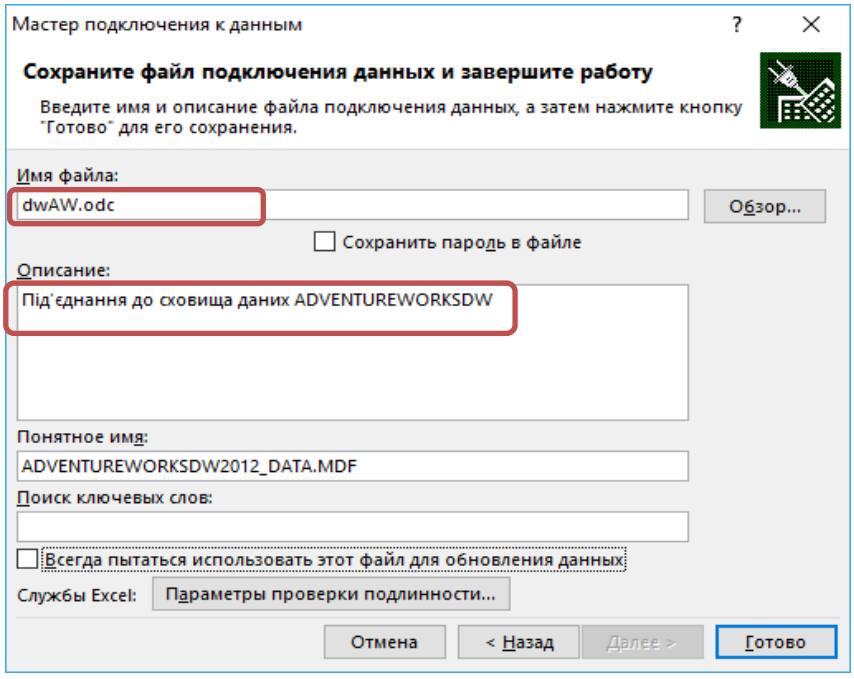


Рис. 6.7. **Ім'я файла та його опис у вікні Майстра під'єднання до даних**

9. Увімкніть прапорець **Разрешить выбор нескольких таблиц** у вікні **Выбор таблицы**. Потім увімкніть прапорці перед такими таблицями: DimCustomer, DimDate, DimGeography, DimProduct, FactInternetSales.

Після цього клацніть кнопку **ОК** (рис. 6.8).

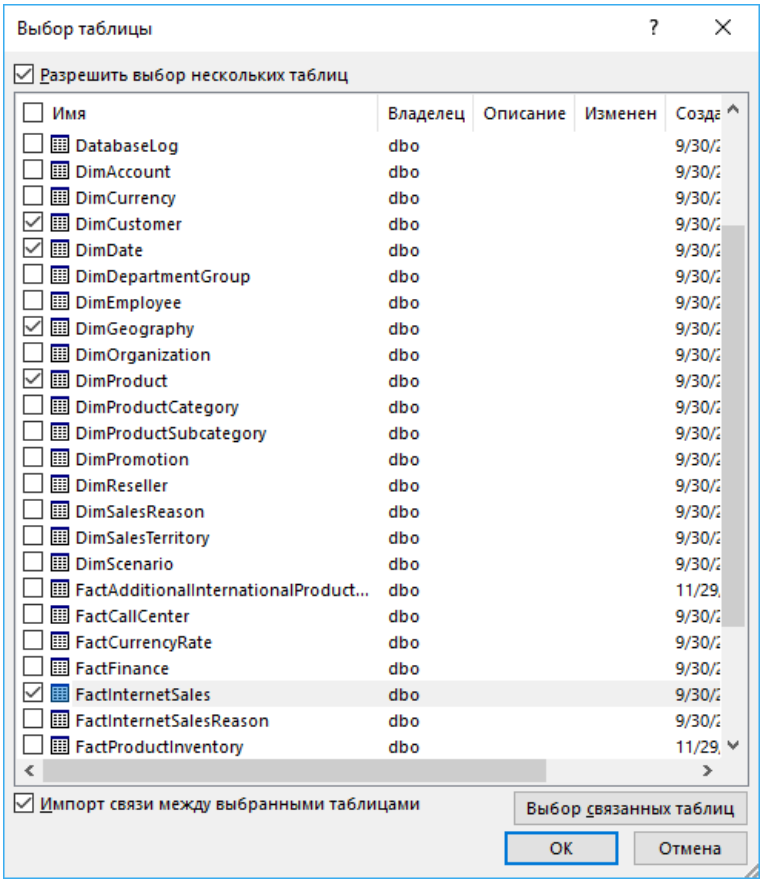


Рис. 6.8. **Вибір таблиць, що будуть використовуватися для аналізу даних**

10. Виберіть перемикач **Отчет сводной таблицы** та клацніть кнопку **ОК** у вікні **Импорт данных** (рис. 6.9).

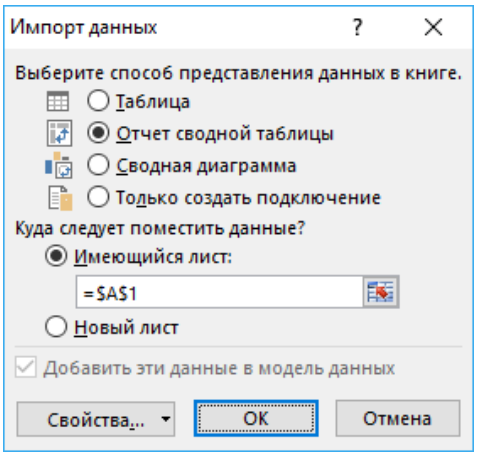


Рис. 6.9. **Вікно Импорт данных**

11. Дочекайтеся, поки на аркуші ***Моніторинг продажів*** з'явиться конструктор зведених таблиць. За його допомогою у подальшому будуватимуться зведені таблиця і діаграма (рис. 6.10).

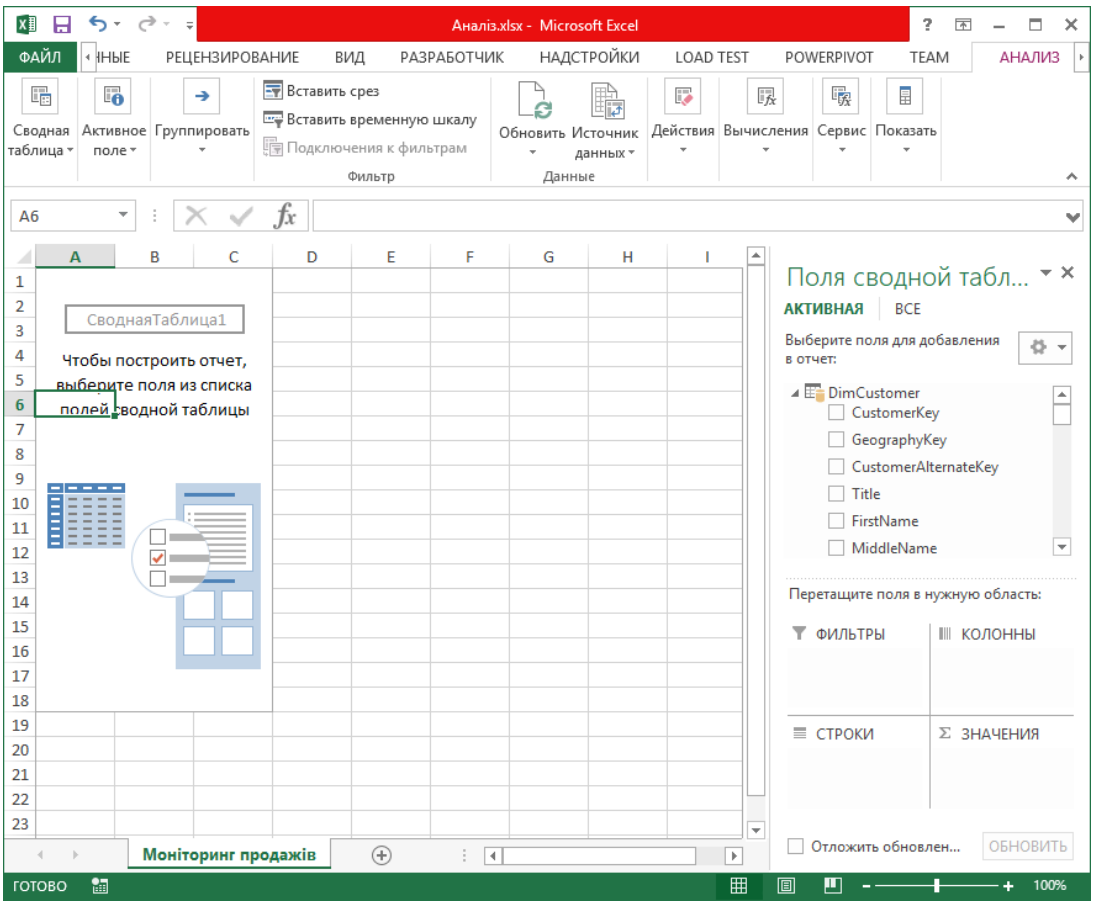


Рис. 6.10. **Конструктор зведених таблиць**

Завдання 2

Створіть зведену таблицю для моніторингу продажів.

Виконання

Для виконання завдання необхідно дотримуватись такої послідовності дій.

1. Перебуваючи на аркуші ***Моніторинг продажів***, перетягніть потрібні поля з верхньої частини панелі **Поля сводной таблицы** у потрібні області нижньої частини цієї панелі згідно з табл. 6.2.

Таблиця 6.2

Розташування полів зведеної таблиці

Таблиці	Поля	Область
DimDate	CalendarYear	СТРОКИ
	MonthNumberOfYear	СТРОКИ
DimGeography	EnglishCountryRegionName	КОЛОННЫ
	City	КОЛОННЫ
DimProduct	EnglishProductName	ФИЛЬТРЫ
FactInternetSales	TotalProductCost	ЗНАЧЕНИЯ

На рис. 6.11 подано панель **Поля сводной таблицы** зі встановленими полями.

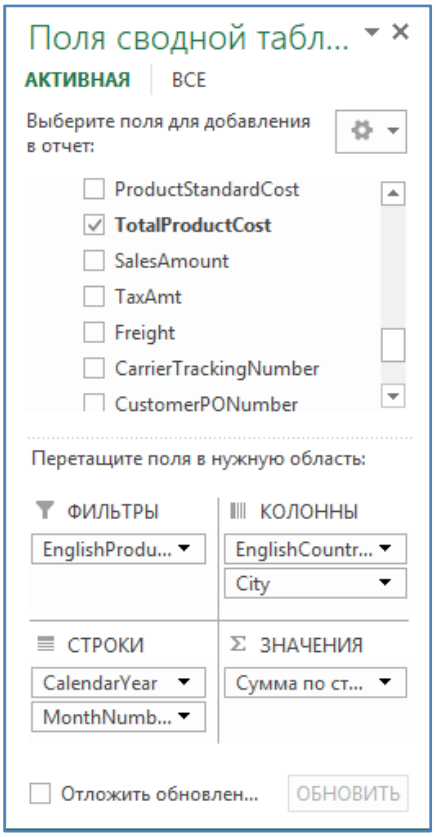


Рис. 6.11. **Панель Поля сводной таблицы зі встановленими полями**

Зліва від панелі з'явилася зведена таблиця (рис. 6.12). Перегляньте її дані. Зверніть увагу на те, що дані у стовпцях згруповані за двома рівнями: країнами та містами, а дані в рядках – за двома рівнями: роками та місяцями.

EnglishProductName	All											
Сумма по столбцу TotalProductCost	Названия столбцов											
	Australia											
Названия строк	Bendigo	Brisbane	Caloundra	Cloverdale	Coffs Harbour	Cranbourne	Darlinghurst	East Brisbane	Findon	Geelong	Gold Coast	Goulburn
2005	**24717,6514**	**21439,7422**	**17923,4464**	**10050,9316**	**24024,1827**	**22006,895**	**23351,8966**	**14687,473**	**8979,1298**	**25004,9112**	**24332,1957**	**39650,4484**
7	12236,2858	2171,2942	2171,2942		413,1463	2171,2942	2171,2942	5722,4032			413,1463	11010,4775
8		4342,5884	4342,5884	1912,1544	4342,5884	4069,3886	2171,2942	4069,3886	2584,4405	5995,603	2171,2942	8685,1768
9	2171,2942		2584,4405		4342,5884	4342,5884	4342,5884		1912,1544	6513,8826	6653,8291	4755,7347
10	4069,3886	8685,1768		1898,0944	4342,5884	2325,3007	6240,6828	1898,0944	2171,2942	2171,2942	4083,4486	4342,5884
11	2171,2942	4342,5884	4342,5884	2171,2942	2171,2942		4083,4486	2584,4405		8426,037	4083,4486	2171,2942
12	4069,3886	1898,0944	4482,5349	4069,3886	8411,977	9098,3231	4342,5884	413,1463	2311,2407	1898,0944	6927,0289	8685,1768
2006	**39911,823**	**47412,3663**	**48102,7932**	**30242,1148**	**32305,3346**	**24666,7194**	**21138,3299**	**31010,4426**	**29188,5932**	**37914,0389**	**32139,9366**	**37703,5419**
1	6513,8826		1898,0944	4069,3886		4496,5949	4342,5884	2171,2942	2171,2942	2171,2942	4083,4486	2171,2942
2	6513,8826		2171,2942		2171,2942	4342,5884	6513,8826	2171,2942	2171,2942	4342,5884		2171,2942
3	2171,2942	8152,8372	4069,3886	4342,5884	2171,2942		1912,1544		4342,5884	6254,7428	6653,8291	4755,7347
4	2171,2942	6240,6828	4342,5884	2171,2942	10737,2777	4342,5884	1912,1544	6254,7428	6513,8826	4069,3886		4482,5349
5		6667,8891	5981,543	4342,5884	4083,4486			2171,2942	2171,2942	1912,1544	2171,2942	5694,2832
6	5708,3432	10338,1914	8138,7772	2171,2942		4342,5884		7066,9754	1898,0944	4636,5414	4342,5884	
7	1518,7864	5265,964	3945,2802		1117,8559		1320,6838	1807,3904	2235,7118	4031,0563	3128,0742	4358,2566
8	5363,786	2641,3676	5950,7732	1926,333	2223,6659	2624,5964		1807,3904	486,7066	1926,333		5464,0666
9			1604,5625	1320,6838	2925,2463		1320,6838	486,7066	1518,7864	486,7066	7197,7268	3445,1194
10	2925,2463	1518,7864		5063,1361	2641,3676	1807,3904	2698,3266	5265,964	2641,3676	5051,0902	1117,8559	2722,4184
11	973,4132		2899,7462	486,7066		1604,5625					1518,7864	1320,6838
12	6051,8943	6586,6478	7100,7453	4348,1015	4233,8842	1105,81	1117,8559	1807,3904	3037,5728	3032,143	1926,333	1117,8559
2007	**68123,8336**	**50225,5465**	**56183,3306**	**49656,0742**	**43332,5184**	**38558,3192**	**25160,8628**	**35771,281**	**44305,3084**	**53592,9292**	**50970,3507**	**58676,6279**
1	4930,7393	2426,4938	2636,6423	3556,3956	2913,2004		1105,81	3111,303	2839,4702	3111,303	5037,636	2839,4702
2	2438,5397	8389,3129	4431,9868	6348,1647	2841,361	1320,6838			3556,3956	2426,4938	8384,5876	2641,3676

Моніторинг продажів

Рис. 6.12. **Зведена таблиця**

2. Укрупніть подання даних у зведеній таблиці, щоб дані відображалися тільки за країнами та роками. Для цього клацніть правою клавішею миші на назві будь-якої країни (наприклад, Australia) і з контекстового меню виберіть команду **Свернуть/Развернуть – Свернуть все поля**. Потім аналогічні дії виконайте для назви року. На рис. 6.13 подано зведену таблицю зі згорнутими даними.

	A	B	C	D	E	F	G	H
1	EnglishProductName	All						
2								
3	Сумма по столбцу TotalProductCost	Названия столбцов						
4		Australia	Canada	France	Germany	United Kingdom	United States	Общий итог
5	Названия строк							
6	2005	780478,0437	88118,6533	108108,2606	142211,4949	174339,1921	661512,1419	1954767,787
7	2006	1274424,917	373871,7428	306757,9856	310869,6938	350778,4984	1266790,537	3883493,375
8	2007	1787464,474	302837,5866	602744,6969	625392,7704	765690,736	1634196,912	5718327,175
9	2008	1532778,074	383095,3782	540142,0494	628467,6141	710413,0064	1926309,118	5721205,24
10	Общий итог	5375145,508	1147923,361	1557752,993	1706941,573	2001221,433	5488808,708	17277793,58

Рис. 6.13. **Зведена таблиця зі згорнутими даними**

3. Відберіть дані у зведеній таблиці за певними товаром. Для цього клацніть на кнопці зі стрілкою донизу в області фільтрів, що розташована ліворуч над зведеною таблицею. Розкрийте вузол **All** і виберіть потрібний товар (наприклад, ***AWC Logo Cap***), а потім клацніть кнопку **ОК**. На рис. 6.14 подано результат фільтрації, з якого випливає, що продажі велосипедів цієї моделі здійснювалися тільки два останні роки, причому продажі збільшувалися в усіх країнах.

	A	B	C	D	E	F	G	H
1	EnglishProductName	AWC Logo Cap						
2								
3	Сумма по столбцу TotalProductCost	Названия столбцов						
4		Australia	Canada	France	Germany	United Kingdom	United States	Общий итог
5	Названия строк							
6	2007	1169,8687	595,3178	678,3854	740,6861	1017,5781	1924,3994	6126,2355
7	2008	1765,1865	1079,8788	962,1997	1176,791	1266,7809	2782,7646	9033,6015
8	Общий итог	2935,0552	1675,1966	1640,5851	1917,4771	2284,359	4707,164	15159,837

Рис. 6.14. **Зведена таблиця з відфільтрованими даними**

4. Поверніться до відображення даних щодо всіх товарів. Для цього клацніть на кнопці зі значком фільтра в області фільтрів, що розташована

ліворуч над зведеною таблицею; виберіть значок **All**, а потім клацніть кнопку **ОК**. Таблиця отримає вигляд, який подано на рис. 6.13.

5. Збережіть зміни, що зроблені у книзі *Excel*.

Завдання 3

Додайте зведену діаграму для моніторингу продажів, щоб вона синхронізувалася із зведеною таблицею.

Виконання

Для виконання завдання необхідно дотримуватись такої послідовності дій.

1. Перебуваючи на аркуші ***Моніторинг продажів***, клацніть будь-яку клітинку зведеної таблиці; у вкладці **Анализ** виберіть значок **Сводная диаграмма** у групі **Сервис**.

2. Виберіть тип діаграми **График** у списку діаграм вікна **Вставка диаграммы** та клацніть кнопку **ОК** (рис. 6.15).

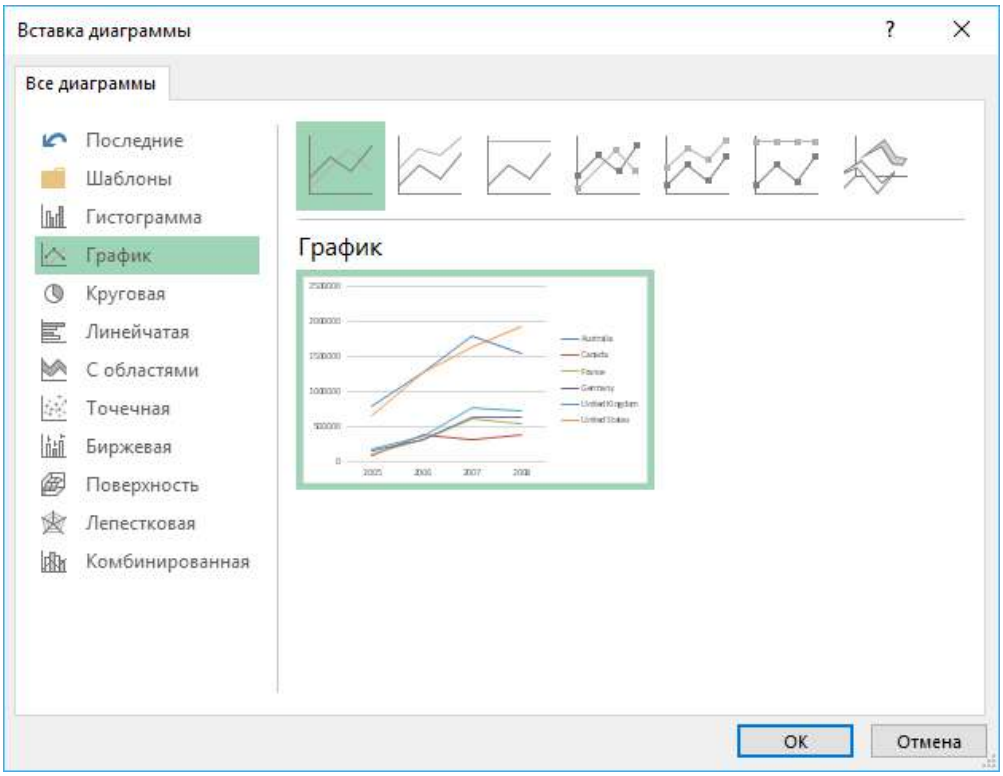
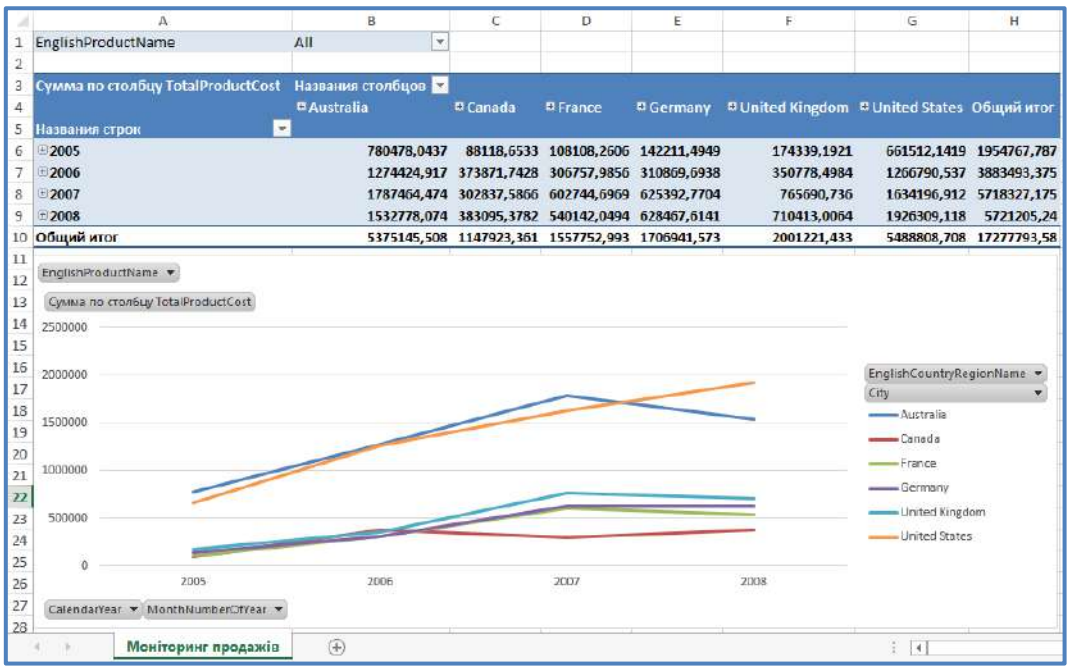


Рис. 6.15. **Вибір типу діаграми**

3. Перетягніть отриману діаграму під зведену та пропорційно збільште її розміри, щоб ширина діаграми співпадала з шириною зведеної таблиці (рис. 6.16).



EnglishProductName	All						
Сумма по столбцу TotalProductCost	Названия столбцов						
	Australia	Canada	France	Germany	United Kingdom	United States	Общий итог
Названия строк							
2005	780478,0437	88118,6533	108108,2606	142211,4949	174339,1921	661512,1419	1954767,787
2006	1274424,917	373871,7428	306757,9856	310869,6938	350778,4984	1266790,537	3883493,375
2007	1787464,474	302837,5866	602744,6969	625392,7704	765690,736	1634196,912	5718327,175
2008	1532778,074	383095,3782	540142,0494	628467,6141	710413,0064	1926309,118	5721205,24
Общий итог	5375145,508	1147923,361	1557752,993	1706941,573	2001221,433	5488808,708	17277793,58

Рис. 6.16. **Зведені таблиця і діаграма**

4. Перевірте взаємодію зведених таблиці та діаграми, вибираючи різні рівні деталізації за роками та країнами, а також фільтрацію за товарами.

У звіті з лабораторної роботи подайте скриншоти (не менше трьох) з результатами експериментів. До кожного скриншоту додайте бізнес-висновок щодо продажів товарів.

5. Поверніть зведені таблицю та діаграму у попередній стан, щоб вони мали вигляд, який подано на рис. 6.16.

6. Збережіть зміни, що зроблені у книзі *Excel*.

Завдання 4

Додайте панель зрізу, щоб надати можливість оперативно прослідковувати дані щодо продажу товарів за їхнім кольором.

Виконання

Для виконання завдання необхідно дотримуватися такої послідовності дій.

1. Перебуваючи на аркуші ***Моніторинг продажів***, клацніть правою клавішею миші заголовок стовпця **A** книги Excel і в контекстовому меню виберіть команду **Вставить**.

2. Збільште ширину стовпця **A** до 23 пунктів (приблизно 4 см). Тут у подальшому буде розташовуватися панель зрізу.

3. Клацніть будь-яку клітинку зведеної таблиці, у вкладці **Анализ** виберіть команду **Вставить срез** у групі **Фильтр**.

4. Виберіть поле ***Color*** у таблиці ***DimProduct***, що знаходиться у вікні **Вставка срезов**, і клацніть кнопку **ОК** (рис. 6.17).

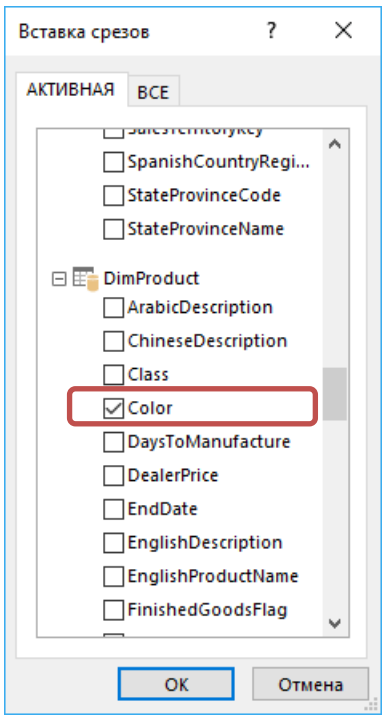


Рис. 6.17. **Вибір поля фільтрації**

5. Перетягніть панель зрізу в стовпець **A** (рис. 6.18).

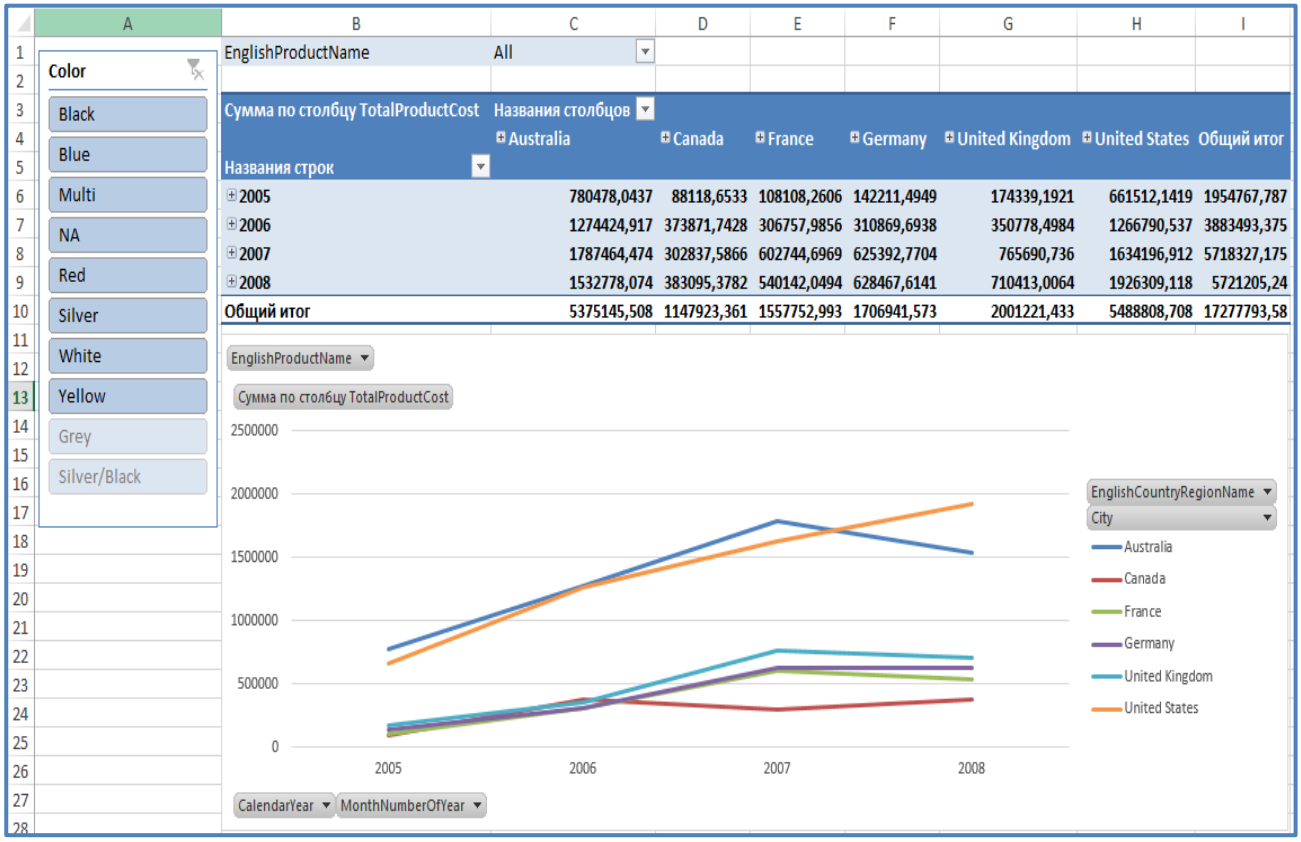


Рис. 6.18. **Зведені таблиця і діаграма з панеллю зрізу**

6. Перевірте роботу панелі зрізу на різних значеннях кольору товарів.

У звіті з лабораторної роботи подайте скриншоти (не менше трьох) з результатами експериментів. Порівняйте динаміку продажів товарів червоного та інших кольорів.

6.3. Проектування сховища даних

Завдання

На основі схеми індивідуальної оперативної бази даних, що отримана у лабораторній роботі 4, побудуйте схему сховища даних.

Основні ідеї

У лабораторній роботі 1 розглядалася база даних ***Хліб*** зі схемою, що подано на рис. 6.19.

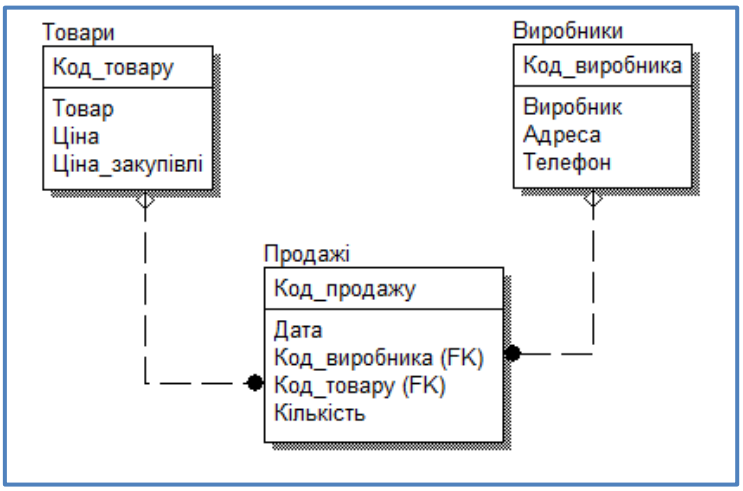


Рис. 6.19. **Схема оперативної бази даних *Хліб***

Відповідне сховище даних ***ХлібСД*** матиме схему, що подана на рис. 6.20. Літери ***СД*** у кінці імені є скороченнями від слів *сховище даних*.

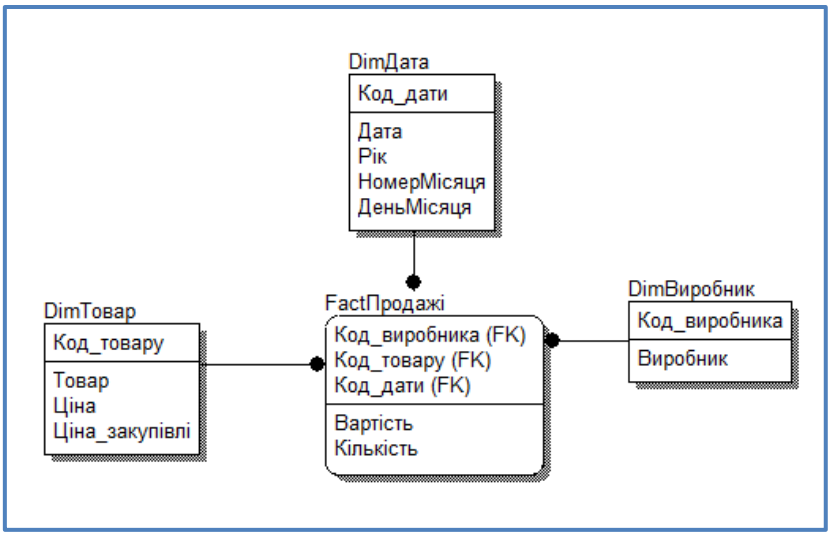


Рис. 6.20. **Схема сховища даних *ХлібСД***

Виконання

Самостійно спроектуйте схему сховища даних на основі індивідуальної оперативної бази даних, що використовувалася у лабораторній роботі 4.

У звіті з лабораторної роботи подайте обидві схеми – оперативної бази даних і сховища даних.

6.4. Створення і заповнення сховища даних

Завдання 1

Додайте до проекту mdf-файл індивідуального сховища даних, потім на основі схеми індивідуального сховища даних запишіть SQL-скрипти для створення і заповнення таблиць вимірів і виконайте їх.

Основні ідеї

Оскільки операції з базами даних виконуватимуться у середовищі *Visual Studio*, спочатку створюють проект Windows Forms з ім'ям ***appData***.

Потім, використовуючи візуальні засоби Visual Studio, створюють mdf-файл. Для сховища даних ***ХлібСД*** він має ім'я ***ХлібСД.mdf***. На рис. 6.21 подано вікно Server Explorer з оперативною базою даних і сховищем даних.

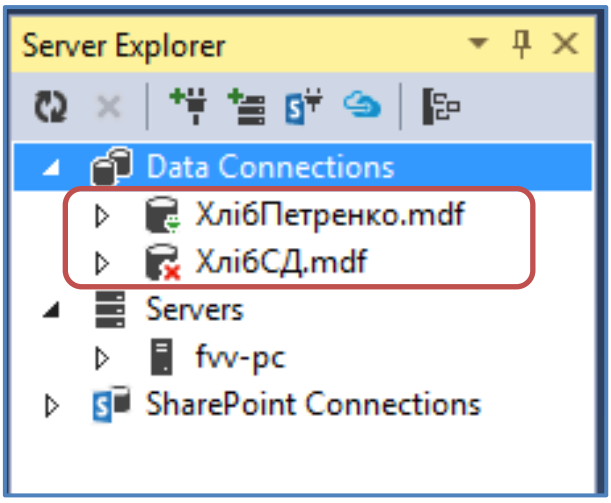


Рис. 6.21. **Оперативна база даних і сховище даних у вікні Server Explorer**

У сховищі даних ***ХлібСД*** є три таблиці вимірів – ***DimТовар***, ***DimВиробник*** і ***DimДата***. Дві перших, характерні для предметної області, заповнюються даними з оперативної бази даних; третя – майже однакова для усіх сховищ. Вона заповнюється виходячи з періоду, за який виконують аналіз даних (у т. ч. і в майбутньому). У поточному завданні розглядається робота тільки з першими двома таблицями.

Створення таблиці ***DimТовар*** здійснюється за допомогою такого скрипта:

```
CREATE TABLE [DimТовар] (
    [Код_товару]    INT          IDENTITY (1, 1) NOT NULL,
    [Товар]         NVARCHAR (25) NULL,
    [Ціна]          MONEY        NULL,
    [Ціна_закупівлі] MONEY        NULL,
    CONSTRAINT [PK_DimТовар] PRIMARY KEY CLUSTERED ([Код_товару] ASC)
);
```

Примітка: запит виконують через під'єднання до сховища даних.

Таблицю ***DimТовар*** заповнюють даними з відповідної таблиці ***Товари***, що знаходиться в оперативній базі даних ***Хліб***. Для цього використовують такий скрипт:

```
-- Встановлення дозволу вставляти явні значення в стовпець ідентифікаторів таблиці
SET IDENTITY_INSERT [Шлях до сховища даних].dbo.DimТовар ON;

-- Заповнення таблиці DimТовар даними з таблиці Товари,
--- що знаходиться в оперативній базі даних Хліб
INSERT INTO [Шлях до сховища да-
них].dbo.DimТовар(Код_товару,Товар,Ціна,Ціна_закупівлі)
SELECT Код_товару, Товар, Ціна, Ціна_закупівлі FROM Товари;

-- Скасування дозволу вставляти явні значення в стовпець ідентифікаторів таблиці
SET IDENTITY_INSERT [Шлях до сховища даних].dbo.DimТовар OFF;
```

Примітки:

1) значенням параметра [*Шлях до сховища даних*] є значення властивості **Primary File Path**, що відображається у вікні властивостей, якщо клацнути значок сховища даних у вікні **Server Explorer** (рис. 6.22);

2) запит виконують під'єднанням до оперативної бази даних.

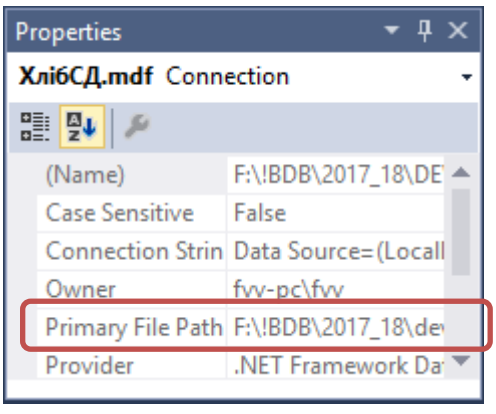


Рис. 6.22. **Властивість Primary File Path**

Створення таблиці ***DimВиробник*** здійснюється за допомогою такого скрипта

```
CREATE TABLE [DimВиробник] (
   [Код_виробника] INT         IDENTITY (1, 1) NOT NULL,
   [Виробник]      NVARCHAR (20) NULL,
   CONSTRAINT [PK_DimВиробник] PRIMARY KEY CLUSTERED ([Код_виробника]
ASC)
)
```

Примітка: запит виконують під'єднанням до сховища даних.

Таблицю ***DimВиробник*** заповнюють даними з відповідної таблиці ***Виробники***, що знаходиться в оперативній базі даних ***Хліб***. Для цього використовують такий скрипт:

```
-- Встановлення дозволу вставляти явні значення в стовпець ідентифікаторів таблиці
SET IDENTITY_INSERT [Шлях до сховища даних].dbo.DimВиробник ON;

-- Заповнення таблиці DimВиробник даними з таблиці Виробники,
-- що знаходиться в оперативній базі даних Хліб
INSERT INTO [Шлях до сховища да-
них].dbo.DimВиробник(Код_виробника,Виробник)
SELECT Код_виробника, Виробник FROM Виробники;

-- Скасування дозволу вставляти явні значення в стовпець ідентифікаторів таблиці
SET IDENTITY_INSERT [Шлях до сховища даних].dbo.DimВиробник OFF;
```

Примітки:

1) значенням параметра [*Шлях до сховища даних*] є значення властивості **Primary File Path**, що відображається у вікні властивостей, якщо клацнути позначку сховища даних у вікні **Server Explorer**;

2) запит виконують під'єднанням до оперативної бази даних.

Виконання

Самостійно додайте до проекту ***appData*** mdf-файл індивідуального сховища даних, потім на основі схеми індивідуального сховища даних запишіть SQL-скрипти для створення і заповнення таблиць вимірів (окрім DimData) та виконайте їх.

У звіті з лабораторної роботи подайте тексти скриптів для створення і заповнення таблиць вимірів, а також сриншоти даних цих таблиць.

Завдання 2

Запишіть SQL-скрипти для створення і заповнення таблиці вимірювань ***DimДата*** та виконати їх.

Основні ідеї

Створення таблиці ***DimДата*** здійснюється за допомогою такого скрипта:

```
CREATE TABLE [DimДата] (
   [Код_дати]   INT     NOT NULL,
   [Дата]       DATETIME NOT NULL,
   [Рік]        INT     NOT NULL,
   [НомерМісяця] INT     NOT NULL,
   [ДеньМісяця]  INT     NOT NULL,
   CONSTRAINT [PK_DimДата] PRIMARY KEY CLUSTERED ([Код_дати] ASC)
);
```

Примітка: запит виконують після під'єднання до сховища даних.

Таблицю ***DimДата*** заповнюють даними залежно від періоду, за який виконують аналіз даних. У поточному завданні розглядається спрощена схема таблиці ***DimДата***. Більш детальну інформацію з цього питання можна знайти за такими посиланнями:

https://www.codeproject.com/Articles/647950/Create-and-Populate-Date-Dimension-for-Data-Wareho;

https://www.mssqltips.com/sqlservertip/4054/creating-a-date-dimension-or-calendar-table-in-sql-server/;

http://ramachandran2010.blogspot.com/2010/06/dimdate-table-script.html.

Для заповнення таблиці ***DimДата*** даними за спрощеною схемою використовують такий скрипт:

```
-- Очищення таблиці DimДата
DELETE from DimДата;

-- Опис змінних для початкової та кінцевої дат
DECLARE @ПочатковаДата DATETIME
DECLARE @КінцеваДата DATETIME

-- Присвоєння значень для початкової та кінцевої дат періоду,
-- в якому буде вестись аналіз (в т. ч. і в майбутньому)
SET @ПочатковаДата = '01/01/2010'
SET @КінцеваДата = '12/31/2017'

-- Використання WHILE-циклу від початкової до кінцевої дат
DECLARE @ПоточнаДата DATETIME
SET @ПоточнаДата = @ПочатковаДата

WHILE @ПоточнаДата <= @КінцеваДата
BEGIN
```

```
 -- Додавання запису у таблицю DimДата
   INSERT INTO DimДата VALUES (
     CONVERT(char(8),@ПоточнаДата,112), -- Ключ
     @ПоточнаДата,
     Year(@ПоточнаДата),
     Month(@ПоточнаДата),
     Day(@ПоточнаДата)
       );
   -- Збільшення поточної дати на 1 день для наступного кроку циклу
   SET @ПоточнаДата = DateAdd(d, 1, @ПоточнаДата)
END;
-- Виведення кількості доданих записів
select count(*) from DimДата;
```

Примітки:

1) значення змінних @ПочатковаДата та @КінцеваДата встановлюють залежно від періоду, за який виконують аналіз даних;

2) до заповнення даними очищають таблицю ***DimДата***, а після заповнення – виводять кількість доданих рядків;

3) запит виконують під'єднанням до сховища даних.

Виконання

Самостійно додайте до індивідуального сховища даних таблицю ***DimData***, а потім заповніть її даними.

У звіті з лабораторної роботи подайте тексти скриптів для створення і заповнення таблиці ***DimData***, а також сриншот даних цієї таблиці.

Завдання 3

На основі схеми індивідуального сховища даних запишіть SQL-скрипти для створення і заповнення таблиці фактів ***FactПродажі*** та виконайте їх.

Основні ідеї

Створення таблиці ***FactПродажі*** здійснюється за допомогою такого скрипта:

```
CREATE TABLE [FactПродажі] (
   [Код_дати]     INT      NOT NULL,
   [Код_виробника] INT      NOT NULL,
   [Код_товару]    INT      NOT NULL,
   [Кількість]     SMALLINT NULL,
   [Вартість]      MONEY    NULL,
   PRIMARY KEY CLUSTERED ([Код_дати] ASC, [Код_виробника] ASC,
[Код_товару] ASC),
   CONSTRAINT [FK_FactПродажі_DimДата] FOREIGN KEY ([Код_дати])
REFERENCES [dbo].[DimДата] ([Код_дати]),
```

```
   CONSTRAINT [FK_FactПродажі_DimВиробник] FOREIGN KEY ([Код_виробника])
REFERENCES [dbo].[DimВиробник] ([Код_виробника]) ON DELETE CASCADE,
   CONSTRAINT [FK_FactПродажі_DimТовар] FOREIGN KEY ([Код_товару])
REFERENCES [dbo].[DimТовар] ([Код_товару]) ON DELETE CASCADE
);
```

Примітка: запит виконують під'єднанням до сховища даних.

Таблицю ***FactПродажі*** заповнюють даними з відповідної таблиці ***Продажі***, що знаходиться в оперативній базі даних ***Хліб***. Сховище даних використовують для аналізу значної кількості даних (десятки тисяч записів і більше). Тому спочатку заповнюють оперативну базу даних (таблицю ***Продажі***) великою кількістю записів. У прикладі використано дані про продажі всіх товарів усіх виробників за період з 01/01/2010 року до 31/12/2017 року. Кількість проданого товару у кожному записі вибирається випадковим чином у межах від 0 до 300 одиниць. У разі, коли Кількість дорівнює нулю, запис не додається до бази даних (товар не продавався). Для цього використовують такий скрипт:

```
--------------------------------
-- Заповнення таблиці Продажі --
--------------------------------

-- Очищення таблиці Продажі
DELETE from Продажі;

-- Опис змінних
DECLARE @ПочатковаДата DATETIME;
DECLARE @КінцеваДата DATETIME;

DECLARE @МінКількість SMALLINT;
DECLARE @МаксКількість SMALLINT;
DECLARE @Кількість SMALLINT;

DECLARE @Код_виробника INT;
DECLARE @Код_товару INT;

-- Присвоєння значень для початкової та кінцевої дат періоду,
-- в якому буде вестись аналіз (в т. ч. і в майбутньому)
SET @ПочатковаДата = '01/01/2010';
SET @КінцеваДата = '12/31/2017';

SET @МінКількість = 0;
SET @МаксКількість = 300;

-- Курсори
DECLARE Виробники_Cursor SCROLL CURSOR FOR
SELECT Код_виробника FROM Виробники;
```

```
DECLARE Товари_Cursor SCROLL CURSOR FOR
SELECT Код_товару FROM Товари;

-- Заповнення даними
OPEN Виробники_Cursor;
OPEN Товари_Cursor;

-- Отримання значення Код_товару першого товару
FETCH FIRST FROM Товари_Cursor INTO @Код_товару;

-- Використання WHILE-циклу від початкової до кінцевої дат
DECLARE @ПоточнаДата DATETIME;
SET @ПоточнаДата = @ПочатковаДата;

WHILE @ПоточнаДата <= @КінцеваДата
BEGIN
   -- Цикл по виробниках
   FETCH FIRST FROM Виробники_Cursor INTO @Код_виробника;
   WHILE @@FETCH_STATUS = 0
   BEGIN
      -- Цикл по товарах
      FETCH FIRST FROM Товари_Cursor INTO @Код_товару;
      WHILE @@FETCH_STATUS = 0
      BEGIN
         -- Обчислення кількості випадковим чином
         SELECT @Кількість = ROUND(((@МаксКількість - @МінКількість -1) * RAND()
            + @МінКількість), 0);

                  -- Щоб не додавалися дані, коли товар не продавався
                  IF @Кількість > 0
          -- Додавання запису у таблицю Продажі
          INSERT INTO Продажі VALUES (
             @ПоточнаДата,
             @Код_виробника,
             @Код_товару,
             @Кількість
          );

         FETCH NEXT FROM Товари_Cursor INTO @Код_товару;;
      END;  -- Цикл по товарах

      FETCH NEXT FROM Виробники_Cursor INTO @Код_виробника;
   END;  -- Цикл по виробниках

   -- Збільшення поточної дати на 1 день для наступного кроку цикла
   SET @ПоточнаДата = DateAdd(d, 1, @ПоточнаДата)
END; -- Цикл по датах
-- Закриття курсорів і звільнення пам'яті
CLOSE Виробники_Cursor;
DEALLOCATE Виробники_Cursor;
```

```
CLOSE Товари_Cursor;
DEALLOCATE Товари_Cursor;

-- Виведення кількості доданих записів
select count(*) from Продажі;
```

Примітка: запит виконують після під'єднання до оперативної бази даних.

Щоб перенести дані з таблиці ***Продажі*** оперативної бази даних у таблицю ***FactПродажі*** сховища даних використовують такий скрипт:

```
-- Очищення таблиці FactПродажі
DELETE FROM [Шлях до сховища даних].dbo.FactПродажі;

-- Перенесення даних з таблиці Продажі у таблицю FactПродажі з обчисленням вартості
INSERT INTO [Шлях до сховища даних].dbo.FactПродажі
        (Код_дати, Код_виробника,Код_товару, Кількість, Вартість)
SELECT CONVERT(char(8),п.Дата,112) AS Код_дати, п.Код_виробника,
        п.Код_товару, п.Кількість,
        т.Ціна*п.Кількість AS Вартість
FROM Товари т
        JOIN Продажі п ON т.Код_товару=п.Код_товару;
```

Примітки:

1) значенням параметра [*Шлях до сховища даних*] є значення властивості **Primary File Path**, що відображається у вікні властивостей, якщо клацнути значок сховища даних у вікні **Server Explorer**;

2) запит виконують після під'єднання до оперативної бази даних.

Виконання

Самостійно додайте до індивідуального сховища даних таблицю фактів. Потім заповніть значною кількістю даних відповідну таблицю в оперативній базі даних. Після цього перенесіть дані з оперативної бази даних у таблицю фактів сховища даних.

У звіті з лабораторної роботи подайте тексти скриптів для створення таблиці фактів у сховищі даних. Заповніть відповідну таблицю в оперативній базі даних, перенесіть дані з оперативної бази даних у таблицю фактів сховища даних. Надайте сриншот даних цієї таблиці.

6.5. Створення зведених таблиць і діаграм у середовищі *Excel* на основі даних сховища

Завдання

На основі індивідуального сховища даних побудуйте панель моніторингу. Дані у ній подайте у вигляді зведеної таблиці та кількох діаграм, а також використайте зріз для швидкої фільтрації даних.

Основні ідеї

Щоб побудувати панель моніторингу на основі даних сховища ***ХлібСД*** потрібно відкрити нову книгу *Excel*, створити файл під'єднання до сховища даних, використавши команду **Данные – Получение внешних данных – Из других источников – Из мастера подключения данных – Дополнительно**.

Потім у конструкторі зведених таблиць побудуйте зведену таблицю, згрупувавши дані в рядках за двома рівнями – роками та місяцями, додайте зведені діаграми та зріз (рис. 6.23).

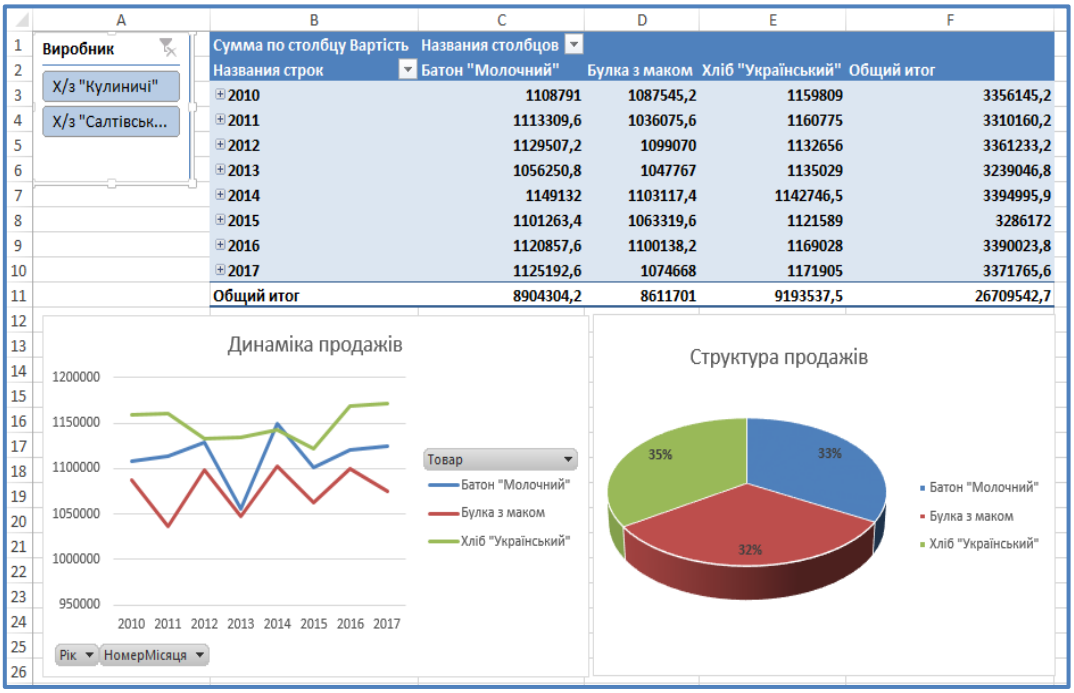


Рис. 6.23. **Панель моніторингу продажів хлібобулочних виробів**

Другу діаграму, яка є незалежною від першої, будують як нову на тому самому файлі під'єднання. Щоб вона реагувала на вибір виробника у панелі зрізу, клацніть правою клавішею миші на панелі зрізу та з контекстового меню виберіть команду **Подключения к отчетам**, а потім у вікні, що відкрилося, додайте посилання на новий звіт зведеної таблиці чи діаграми.

Виконання

Самостійно побудуйте панель моніторингу на основі даних, що зберігаються в індивідуальному сховищі.

У звіті з лабораторної роботи подайте скриншот панелі моніторингу та проаналізуйте з економічної точки зору процеси, що зображені на діаграмах панелі.

Завдання для самостійного виконання

1. Додайте до бази даних ***Хліб*** ще дві таблиці:
Накладні(Код_накладної, Номер_накладної, Дата, Код_виробника);
Товари_накладних(Код_товару_накладної, Код_накладної, Код_товару, Кількість).

Установіть зв'язки між цими та іншими таблицями бази даних.

У новій таблиці заповніть значною кількістю даних про товари, що надходять до хлібного кіоску.

На основі цих таблиць додайте до сховища даних таблицю фактів ***FactНадходження***; заповніть цю таблицю даними на основі даних таблиць ***Накладні*** та ***Товари_накладних***, а потім побудуйте в *Excel* панель моніторингу надходжень.

2. Створіть панель моніторингу прибутку, який визначається за таким виразом:

(Ціна – Ціна_закупівлі) * Кількість.

Підказка: створіть у базі даних подання і використайте його замість таблиці фактів.

3. Створіть панель моніторингу руху товарів, у якій за кожний день відобразіть кількість отриманого та проданого товару та різницю між ними (залишки).

4. Для кожної з наступних задач зробіть копію з папкою проекту та повторіть дії, що описані у п. п. 3, 4, із врахуванням таких обставин:

4.1. кількість проданих товарів визначається не випадковим чином, а змінюється за певними законом (наприклад, за синусоїдою, повний період якої дорівнює одному року);

4.2. облік проданих товарів ведеться за чеками, а не у цілому за день, як описано у п. 3.

5. Виконайте такі дослідження:

5.1. визначте за який максимальний період можна розмістити дані у сховищі виходячи з обсягів доступної пам'яті на диску вашого комп'ютера;

5.2. побудуйте графік залежності обсягу пам'яті, що займає сховище даних від періоду, за який зберігаються дані у ньому;

5.3. визначте як залежить час виконання запитів на побудову зведених таблиць від обсягу сховища даних. Результати подайте у вигляді відповідних діаграм.

Примітка: у звіті за кожним завданням подайте запити на його виконання та скриншот результату їхнього виконання.

Лабораторна робота 7
Виконання CRUD-операцій в базі даних MongoDB

Цілі роботи

1. Набуття практичних навичок встановлення СКБД MongoDB.

2. Набуття практичних навичок роботи в оболонці СКБД MongoDB.

3. Вироблення умінь додавання, зміни та видалення даних у базі даних MongoDB.

4. Отримання практичних навичок побудови запитів на вибірку в документо-орієнтованій базі даних.

5. Вироблення практичних навичок проектування документо-орієнтованої бази даних.

Перед виконанням роботи студент повинен знати:

основні відмінності документо-орієнтованої бази даних від реляційної;

основні команди оболонки MongoDB;

призначення і основні властивості баз даних MongoDB;

структурні елементи бази даних MongoDB і їхні властивості;

основи проектування баз даних MongoDB;

основи побудови запитів до СКБД MongoDB.

Після виконання роботи студент повинен уміти:

проектувати документо-орієнтовані бази даних;

установлювати СКБД MongoDB;

створювати бази даних MongoDB;

виконувати базові операції в оболонці MongoDB;

виконувати CRUD-операції в СКБД MongoDB.

Хід роботи

7.1. Установлення СКБД MongoDB.

7.2. Базові операції в оболонці MongoDB.

7.3. Проектування бази даних.

7.4. Створення бази даних і колекцій документів в ній. Додавання даних.

7.5. Оновлення даних.

7.6. Видалення даних.

7.7. Вибірка даних з бази.

Форма звітності

За результатами виконання роботи оформіть електронний звіт засобами Word. До кожного завдання запишіть формулювання задачі, надайте текст команд і зробіть скриншот з результатами виконання.

У висновках з лабораторної роботи порівняйте засоби виконання CRUD-операцій в реляційних базах даних і MongoDB.

Критерії оцінювання

У 12-бальній системі оцінка результату захисту лабораторної роботи формується за такими правилами.

1. За виконання завдань, що описані в інструкції, може бути виставлено від 0 до 6 балів.

2. За кожне завдання із діапазону 1 – 6 п. "Завдання для самостійного виконання" може бути виставлено від 0 до 1 бала.

3. За завдання 7 п. "Завдання для самостійного виконання" може бути виставлено від 0 до 5 балів.

4. За кілька варіантів розв'язання кожного із завдань із діапазону 1 – 6 п. "Завдання для самостійного виконання" додається 1 бал.

5. За вибір варіанту, який з кількох варіантів розв'язання є оптимальним, та обґрунтування вибору додається 1 бал.

6. За запізнення із захистом лабораторної роботи знімається 2 бала за кожний тиждень.

Отримана кількість балів за відповіді на кожне завдання лабораторної роботи підсумовується. У результаті такого підрахунку студент може отримати від 0 до 12 балів.

Рекомендована література: [7; 9; 17; 23].

Основні поняття

MongoDB (від англ. humongous – величезний) – це крос-платформна СКБД для управління документо-орієнтованою базою даних. Вона є безкоштовною і має відкритий вихідний код. Ця база даних використовує JSON-подібні документи (BSON), не вимагаючи попереднього опису схем таблиць.

MongoDB призначена для гнучкої і дуже швидкої роботи навіть за великих обсягів даних.

Під час проектування цієї СКБД спочатку закладалася висока доступність, підтримка складних динамічних схем і простий розподіл даних на декількох серверах.

Вона має інтуїтивно зрозумілу модель даних і дає можливість подавати розвинені, ієрархічно організовані структури даних у вигляді документа; часто дозволяє уникнути властивих реляційним СКБД складнощів, пов'язаних зі з'єднанням декількох таблиць.

MongoDB найчастіше застосовується в таких сферах:

веб-застосування з високим трафіком;

застосування з управлінням типом контенту;

аналітика в режимі реального часу;

висока швидкість передавання даних (медіа, SaaS – програмне забезпечення як послуга, Gaming);

фінанси;

телекомунікації;

здоров'я.

СКБД MongoDB має такі можливості:

Javascript як мова для формування запитів;

повна підтримка індексів з будь-якого елемента (навіть вкладеного);

реплікація і висока доступність. Віддзеркалення за допомогою локальних і глобальних мереж для масштабування і забезпечення збереженості даних;

авто-шардінг, який підтримує горизонтальне масштабування;

вибірка, в якій потужні запити документів засновані на їхній структурі;

швидкі збереження і зміна даних, що реалізуються через прості атомарні операції;

технологія Map/Reduce, яка забезпечує гнучку агрегацію даних;

технологія GridFS, яка надає змогу використання файлів будь-якого розміру в базі даних без ускладнення застосувань;

комерційна підтримка, яка надає комерційні послуги, безкоштовне навчання і консалтинг;

до недавнього часу були відсутні транзакції. Атомарність в ній гарантується тільки на рівні цілого документа. Відсутнє поняття "ізоляції". Будь-які дані, які зчитуються одним клієнтом, можуть паралельно змінюватися іншим клієнтом;

відсутній оператор "join". Дані подаються денормалізованним способом.

MongoDB орієнтована на роботу із застосуваннями. Існує докладна і якісна документація та велика кількість прикладів і драйверів для таких популярних мов, як:

Java, JavaScript, Node.js;

C, C++, C#;

PHP, Python, Perl;

Ruby, Grails&Groovy;

Go, Erlang.

Приклад 1. *Подання документа в колекції Product бази даних MongoDB.*

```
{
"productNumber": 3,
"nameProduct": "Хліб \"Український\"",
"price": 12,
"purchasePrice": 10.50
}
```

У MongoDB підтримуються такі **типи даних**:

String – рядковий тип даних в кодуванні UTF-8;

Array (масив) – тип даних для зберігання масивів елементів;

Binary data (двійкові дані) – тип для зберігання даних у бінарному форматі;

Boolean – булевий тип даних, який зберігає логічні значення TRUE або FALSE, наприклад: {"married": FALSE};

Date – зберігає дату в форматі часу Unix;

Double – числовий тип даних для зберігання чисел з плаваючою крапкою;

Integer – числовий тип даних для зберігання цілочисельних значень, наприклад: {"age": 29};

JavaScript – тип даних для зберігання коду javascript;

Min key/Max key – використовуються для порівняння значень з найменшим/найбільшим елементом BSON;

Null – тип даних для зберігання значення Null;

Object – рядковий тип даних;

ObjectID – тип даних для зберігання id документа;

Regular expression – застосовується для зберігання регулярних виразів;

Symbol – тип даних, ідентичний рядковому. Використовується для тих мов, в яких є спеціальні символи;

Timestamp – застосовується для зберігання часу.

Між елементами реляційних баз даних і MongoDB є певна відповідність (табл. 7.1).

Таблиця 7.1

Відповідність між RDMS і MongoDB

RDBMS	MongoDB
Table	Collection
Row	Document
Column	Field
Joins	Embedded documents, linking

Головне правило проектування структури даних в базі даних MongoDB – вона повинна підкорятися вимогам програми та бути максимально оптимізованою для найчастіших запитів. Тому в цих базах немає операторів Join. Наприклад, у накладних щодо товарів кожна позиція в замовленні зберігає в собі посилання на продукт і деякі його атрибути (зокрема, назву). Така денормалізація необхідна, щоб не запитувати об'єкт продукту під час вибірки з бази даних інформації про замовлення.

Для роботи з базою даних у цілому використовують такі **оператори**:

db – для відображення бази даних, яка використовується;

use <database> – для встановлення поточної бази даних;

show dbs – для відображення всіх доступних баз даних;

db.stats() – для відображення статистики щодо поточної бази даних;

db.dropDatabase() – для видалення поточної бази даних.

Практична частина

7.1. Установлення СКБД MongoDB

Завдання 1

Установіть СКБД MongoDB на власному комп'ютері.

Виконання

Для виконання завдання необхідно дотримуватись такої послідовності дій.

1. Створіть на диску C: таку папку:

C:\Data\db.

У ній будуть зберігатися бази даних MongoDB.

2. Перейдіть на сайт MongoDB за посиланням https://www.mongodb.com/download-center#community.

3. Клацніть кнопку DOWNLOAD(msi), яка розташована у вкладці Community Server (рис. 7.1).

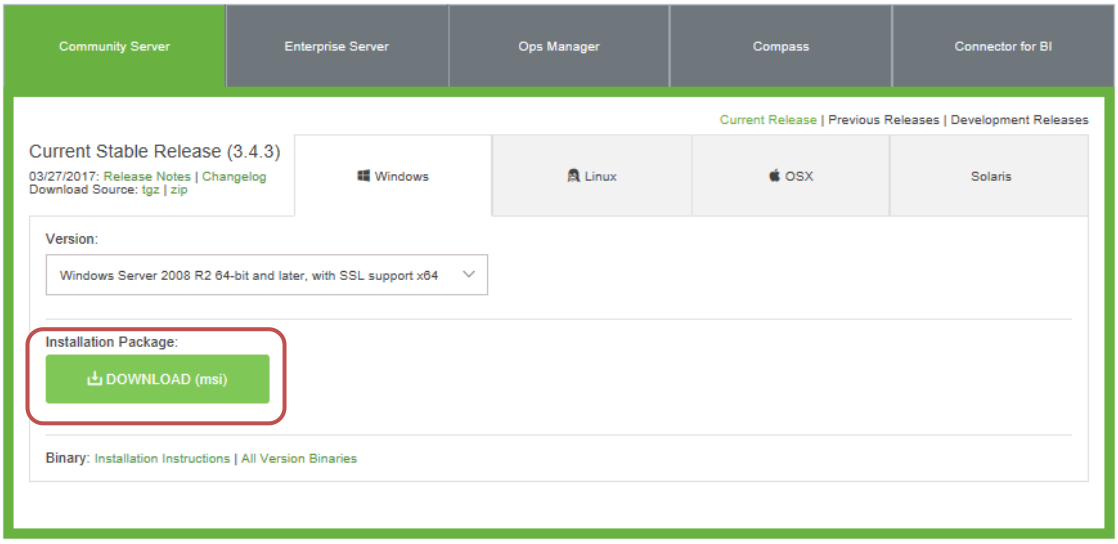


Рис. 7.1. **Кнопка DOWNLOAD(msi) у вкладці Community Server**

4. Відкрийте папку, в якій збережено msi-файл, і двічі клацніть на його позначці, щоб запустити інсталятор СКБД MongoDB.

5. Клацніть кнопку **Next** у двох наступних вікнах установника, потім копку **Complete** в третьому вікні та кнопку **Install** – у четвертому. Дочекайтеся завершення процесу установки та клацніть кнопку **Finish**.

6. Відкрийте папку C:\Program Files\MongoDB\Server\3.4\bin і створіть на робочому столі ярлики для програм mongod (сервер) і mongo (оболонка).

Примітка: першим завжди запускайте сервер, а потім – оболонку.

7.2. Базові операції в оболонці MongoDB

Завдання 2

Ознайомтеся з основними елементами бази даних: список баз даних, поточна база даних, колекція документів, створення бази даних і колекції документів, їхні статистичні дані.

Виконання

Для виконання завдання необхідно дотримуватись такої послідовності дій.

1. Запустіть сервер СКБД MongoDB, двічі клацнувши ярлик програми **mongod**. Відкривається консольне вікно сервера mongod (рис. 7.2). У його останньому рядку зазначено, що сервер підключено до порту 27017. Це порт СКБД MongoDB за замовчуванням.

Надалі в цьому вікні користувач не виконує ніяких операцій, крім, можливо, зупинки сервера за допомогою комбінації клавіш **Ctrl + C**. Зупинити сервер можна, закривши вікно його консолі. У цьому вікні відображаються команди, що отримані сервером і його реакція на них.

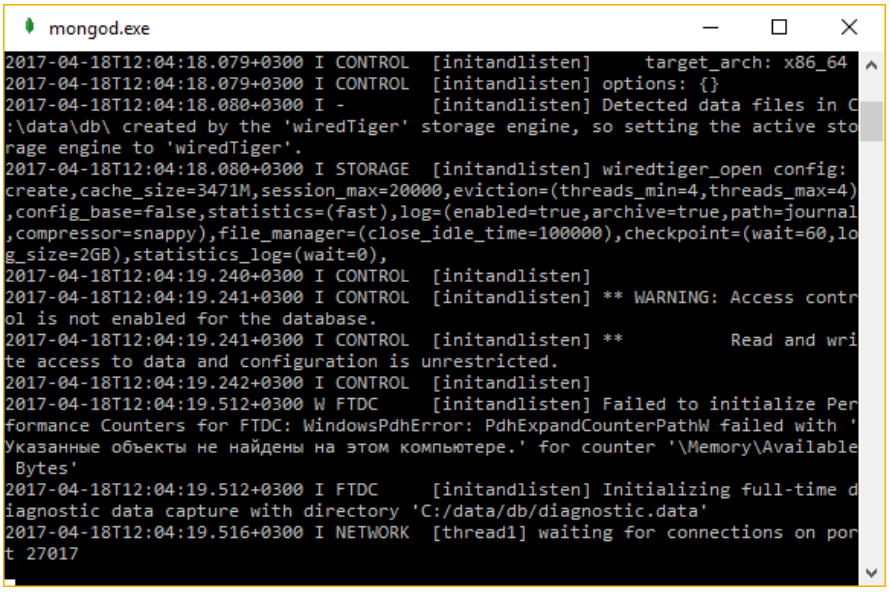


Рис. 7.2. **Консольне вікно сервера mongod**

2. Запустіть оболонку СКБД MongoDB, двічі клацнувши ярлик програми **mongo**. Відкривається консольне вікно оболонки mongo (рис. 7.3). У його першому рядку вказана версія оболонки, а в другому – підключення до бази даних. В останньому рядку відображається символ запрошення **>**. Після нього вводиться команда до сервера.

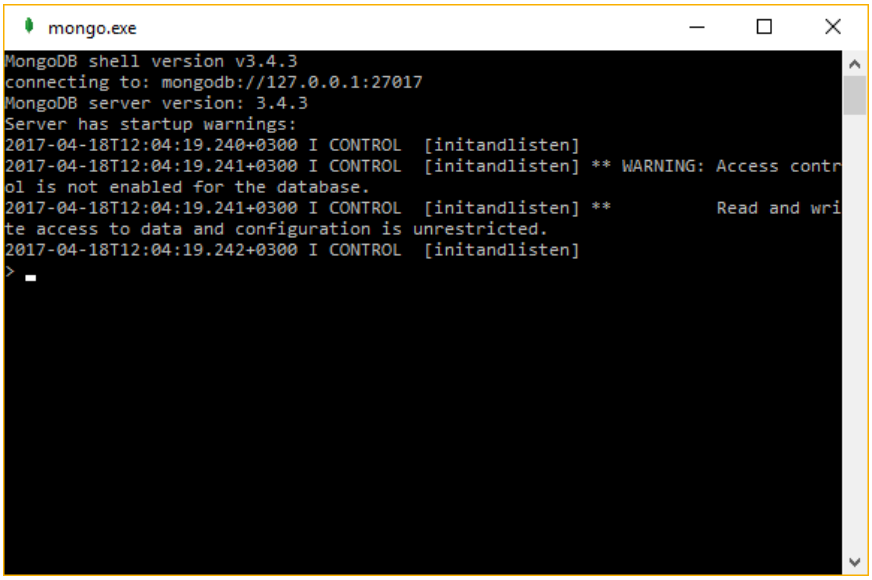


Рис. 7.3. **Консольне вікно оболонки mongo**

Примітка: зазвичай в консолі розташовують команду в одному рядку. Якщо потрібно записати її в кількох рядках, перший рядок закінчують відкривальною дужкою (круглою, квадратною або фігурною). Усі наступні рядки починаються символом "три крапки" (...) як ознакою продовження команди. Команда закінчується після уведення відповідної закривальної дужки.

3. Дізнайтеся ім'я поточної бази даних, ввівши команду

```
Db
```

На рис. 7.4 показано результат виконання цієї команди. Поточною базою даних є база ***test*** (база даних за замовчуванням).

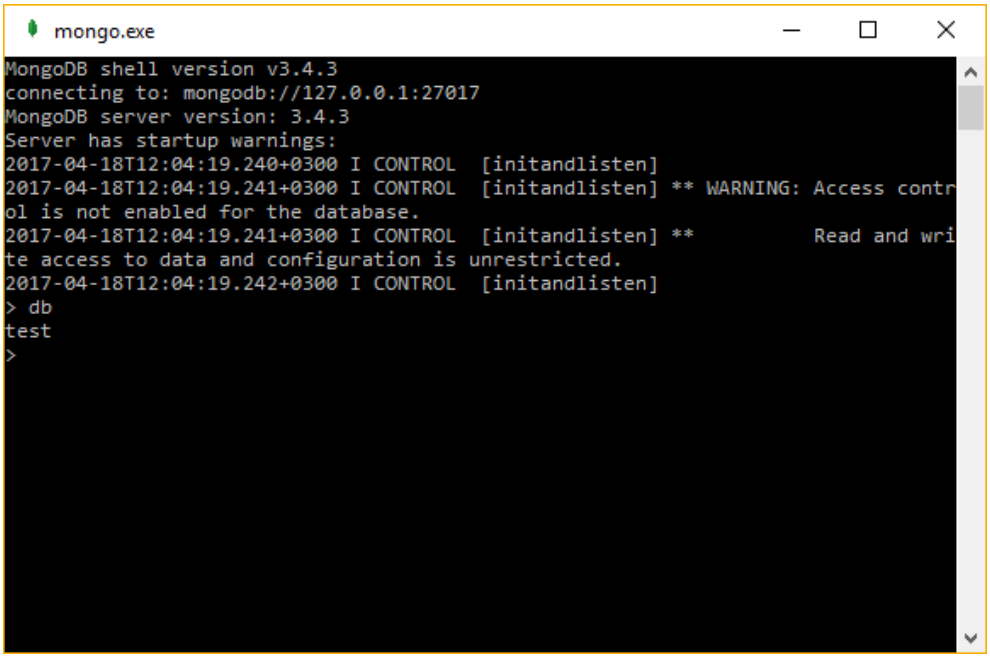


Рис. 7.4. **Визначення поточної бази даних**

4. Щоб визначити імена всіх існуючих баз даних введіть команду

```
show dbs
```

На рис. 7.5 показано результат виконання цієї команди. Є тільки дві системні бази даних – ***admin*** і ***local***. База даних ***test*** ще не існує, оскільки в ній не зберігалися дані.

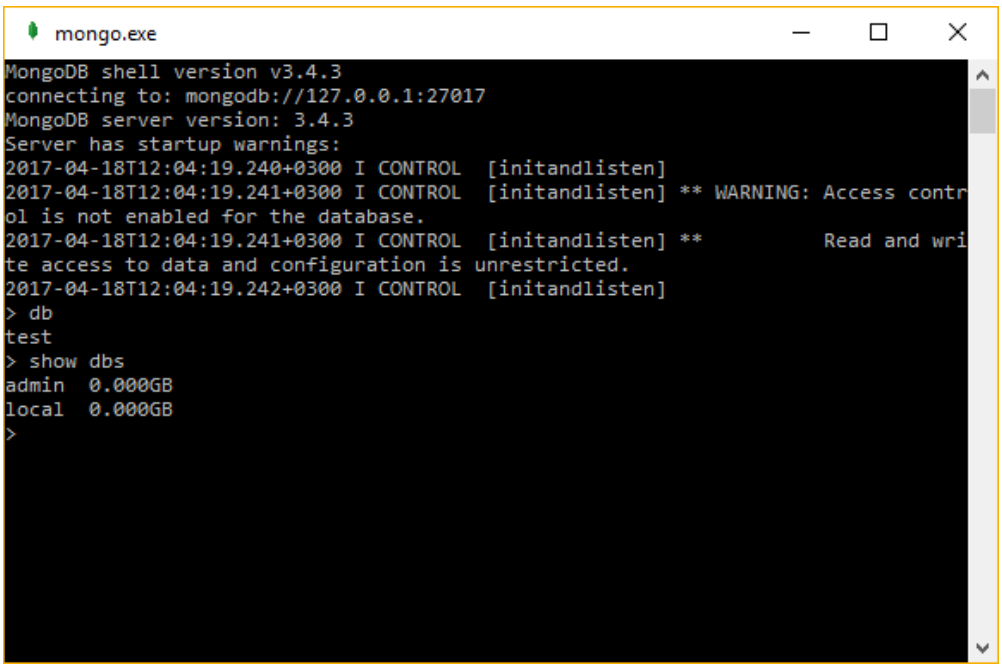


Рис. 7.5. **Визначення імен усіх баз даних**

5. Перевірте можливість введення команд шляхом вставки з буфера. Для цього скопіюйте з інструкції команду ***show dbs*** у буфер. Потім перейдіть у консольне вікно **mongo** та спробуйте вставити в командний рядок вміст буфера.

Якщо спроба буде невдалою, налаштуйте вікно консолі. Для цього:

5.1. клацніть правою клавішею миші на заголовку консольного вікна mongo і виберіть з контекстного меню команду **Властивості**;

5.2. установіть прапорець **Выделение мышью** в групі **Правка** вікна **Свойства** (рис. 7.6);

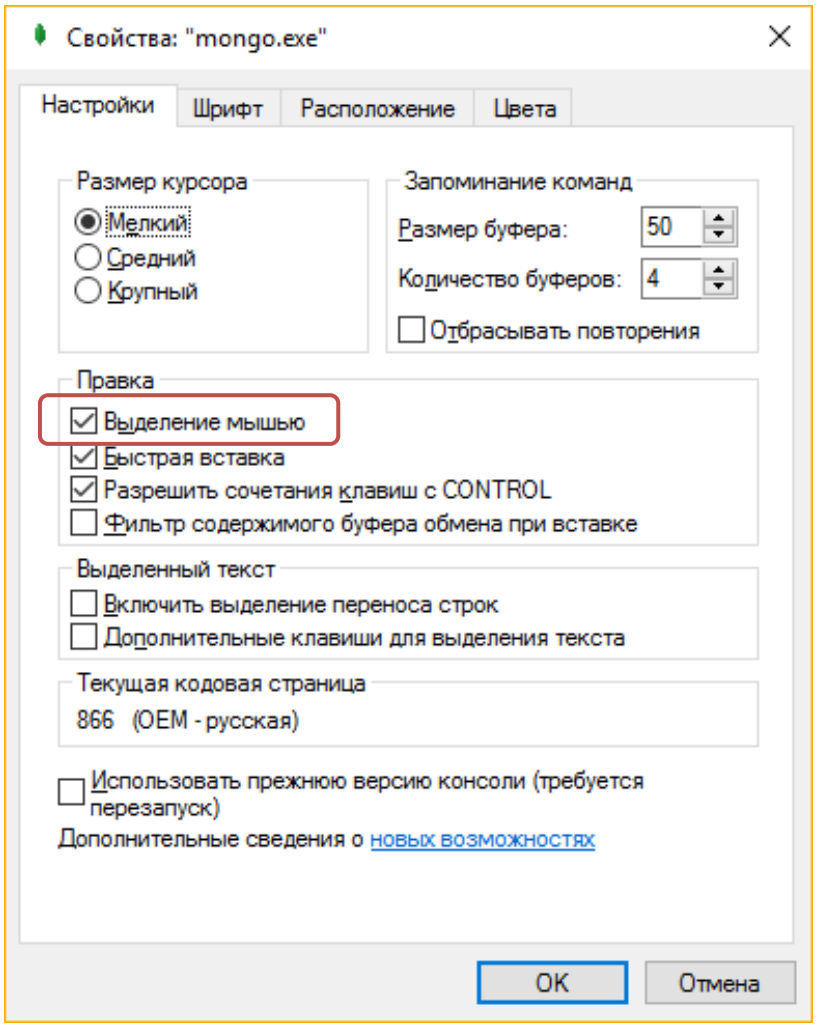


Рис. 7.6. **Прапорець *Выделение мышью* в групі *Правка***

5.3. Клацніть кнопку **ОК**;

5.4. Перевірте можливість вставки тексту з буфера в командний рядок клацанням правої клавіші миші.

Примітка: вставку тексту з буфера в командний рядок можна виконувати без налаштування властивостей шляхом вибору контекстної команди **Змінити – Вставити**.

6. Очистіть вікно **mongo** командою

```
cls
```

7. Вставте документ в неіснуючу колекцію ***students*** неіснуючої бази ***test*** (зараз вона поточна) за допомогою такої команди:

```
db.students.insert({прізвище: "Демченко", курс: 2})
```

На рис. 7.7 показано результат виконання цієї команди.

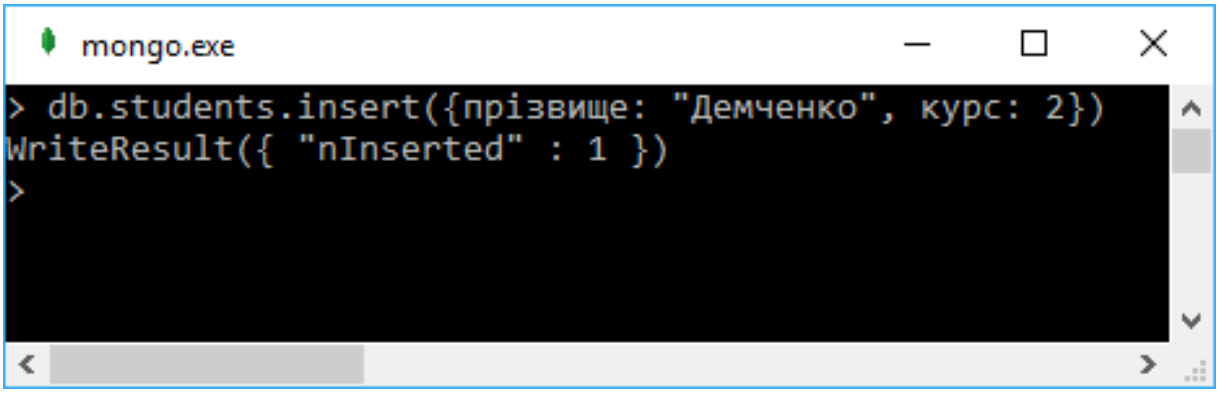


Рис. 7.7. **Введення документа**

8. Перевірте імена існуючих баз даних командою:

```
show dbs
```

На рис. 7.8 показано результат виконання цієї команди. До списку імен додалася база даних ***test***.

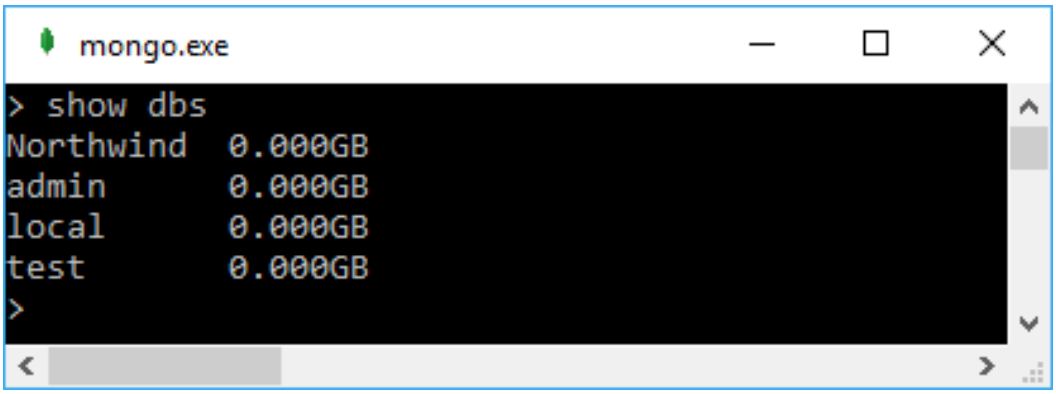


Рис. 7.8. **Існуючі бази даних**

9. Визначте імена всіх колекцій поточної бази даних ***test***, ввівши команду:

```
show collections
```

На рис. 7.9 показано результат виконання цієї команди. У базі даних ***test*** міститься тільки одна колекція ***students***. Таким чином, додавання навіть одного документа в колекцію створює базу даних і колекцію.

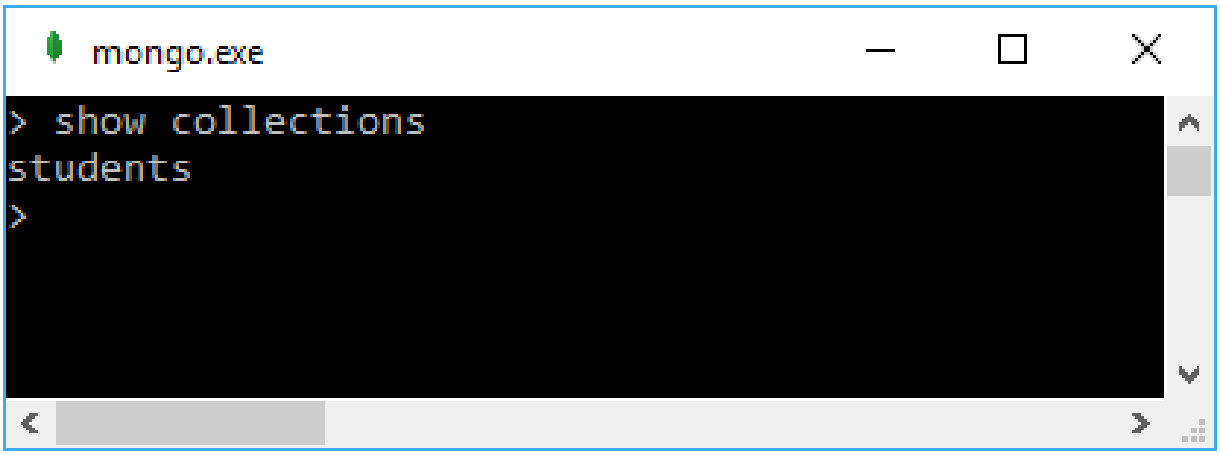


Рис. 7.9. **Колекція поточної бази даних**

10. Перевірте вміст колекції ***students*** за допомогою команди

```
db.students.find()
```

На рис. 7.10 показано результат виконання цієї команди. Колекція складається з одного документа з даними, які введені командою **insert**. До них додано значення ключа документа. Воно створене сервером автоматично, оскільки не було задано явно.

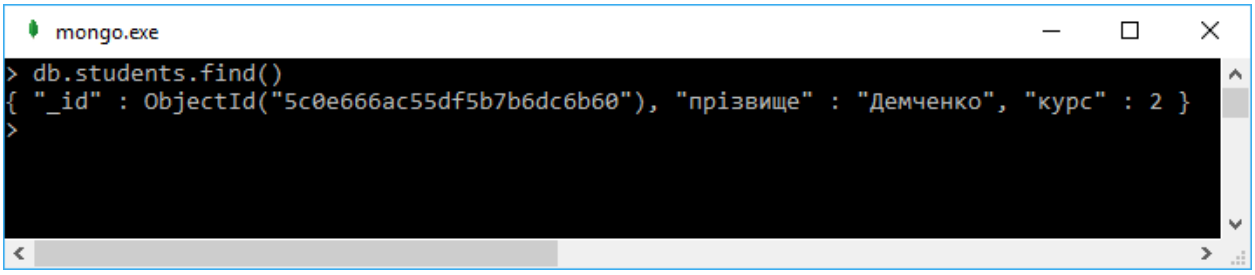


Рис. 7.10. **Вміст колекції *students***

11. Визначте статистичні дані поточної бази даних ***test***, ввівши команду

```
db.stats()
```

На рис. 7.11 показано результат виконання цієї команди. У базі даних ***test*** міститься тільки одна колекція і один документ; указано середній розмір документа і загальний розмір даних, а також один індекс, який створений за ключем колекції.

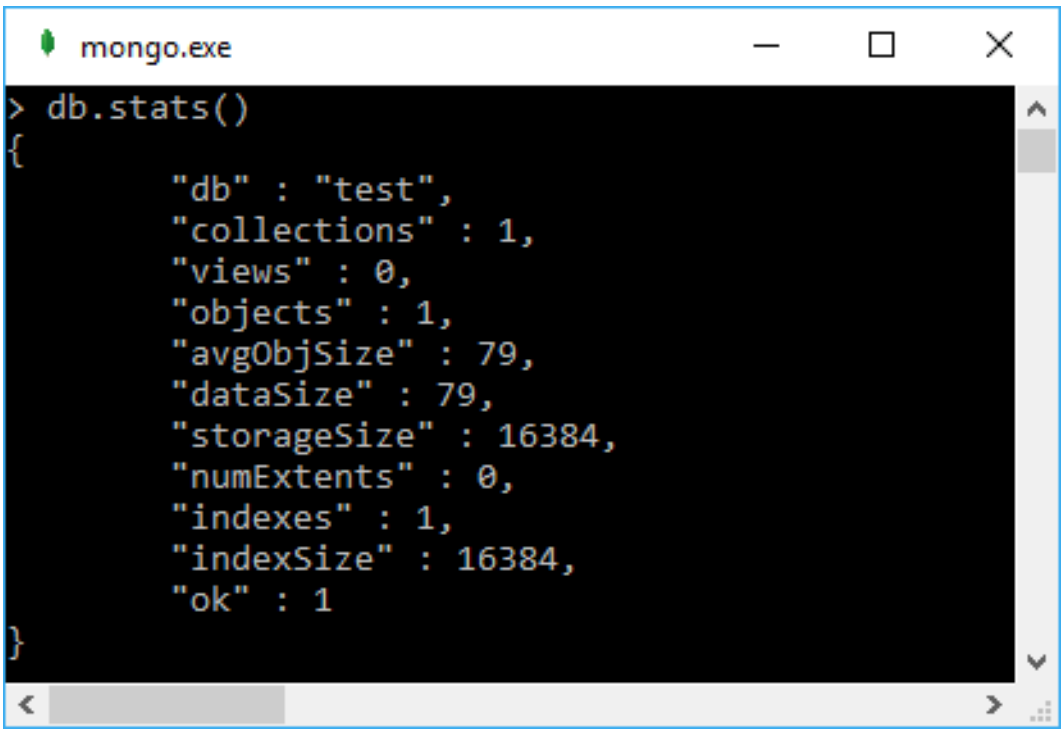


Рис. 7.11. **Статистичні дані бази даних *test***

Аналогічним чином можна дізнатися статистичні дані колекції ***students*** з поточної бази даних, увівши команду

```
db.students.stats()
```

7.3. Проектування бази даних

Завдання 3

Спроектуйте базу даних ***market*** для ведення бізнес-процесу отримання хлібобулочних товарів у кіоску.

Ідея розв'язку

Отримання товарів визначається документом ***Накладна***. Подання його в реляційній базі даних було побудовано та використано на попередніх лабораторних роботах. У даній роботі необхідно перейти від реляційної моделі даних до документо-орієнтованої.

Виконання

У реляційній моделі документ ***Накладна*** подавався схемою, що зображена на рис. 7.12.

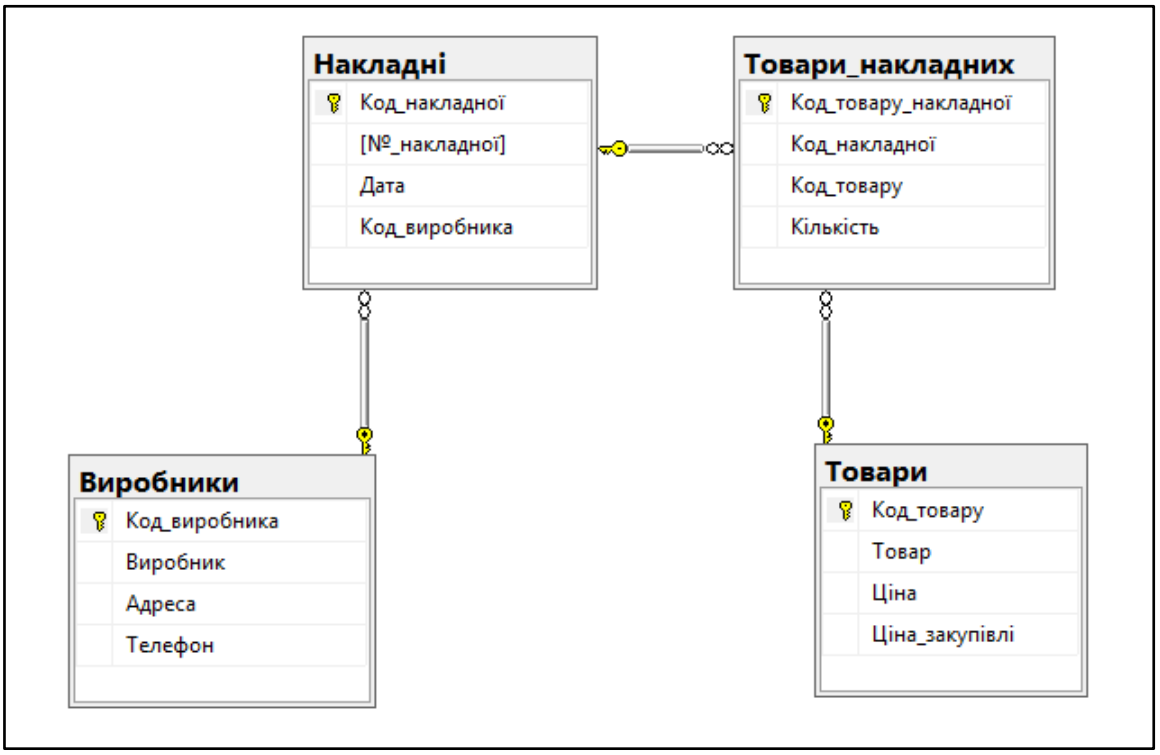


Рис. 7.12. **Схема реляційної моделі документа *Накладна***

Головним принципом побудови документо-орієнтованої моделі є простота більшості запитів до бази даних. Бажано, щоб у запитах не використовувалися дані з документів, що містяться в інших колекціях, оскільки в документо-орієнтованих базах даних відсутня операція Join. Зауважимо, що саме вона в більшості випадків уповільнює виконання запитів у реляційних базах даних.

Процес отримання товарів у бізнес-процесі описується документом ***Накладна***, а більшість запитів, пов'язаних з цим процесом, зводиться до роботи з даними цього документа. Тому бажано в базі даних зберігати всі дані накладної в одному місці. Отже, документ ***Накладна*** в документо-орієнтованій моделі можна подати схемою, що зображена на рис. 7.13.

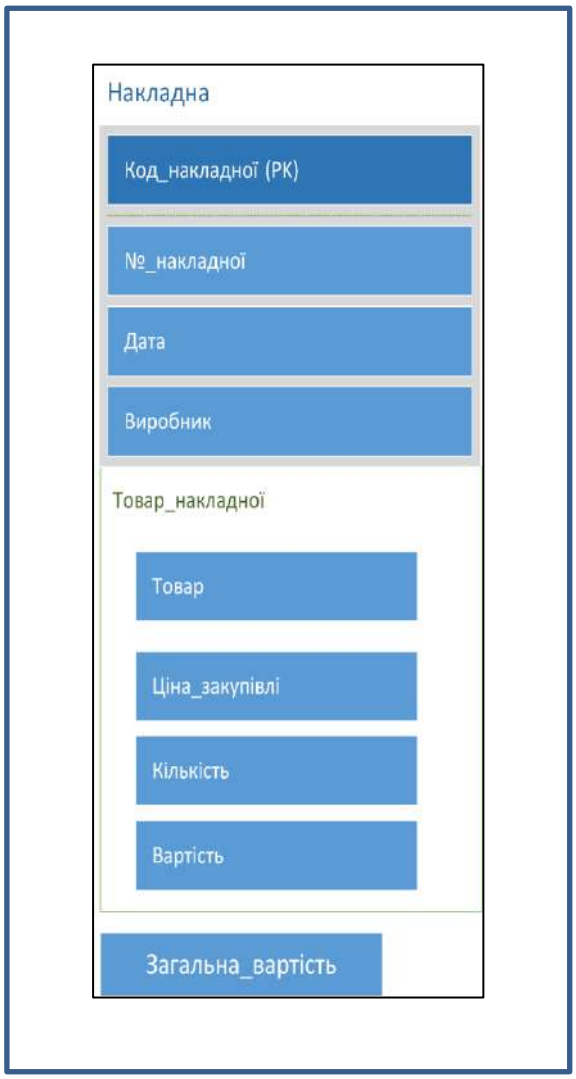


Рис. 7.13. **Схема документа *Накладна* в документо-орієнтованій моделі**

На схемі, що подана на рис. 7.13, показано, що документ ***Накладна*** містить вкладений документ ***Товар_накладної***. Оскільки в одній накладній, як правило, містяться дані за кількома отриманими товарами, створюється масив документів ***Товар_накладної***. Також зауважимо, що в схемі є обчислювані поля ***Вартість*** і ***Загальна_вартість***, щоб не виконувати обчислення в кожному запиті. Це робить модель ненормалізованою, але значно прискорює швидкість виконання запитів.

На рис. 7.14 подано схему документа ***Накладна*** в форматі квазі-JSON. У ній є такі відмінності від схеми, що зображена на рис. 7.13:

усі елементи записано англійською мовою, щоб спростити набір тексту запитів;

поле ***Виробник*** подано вкладеним документом ***manufacturer***, оскільки крім назви виробника в деяких запитах може знадобитися і його номер. З цієї ж причини додано номер товару у вкладений документ ***product***;

колекція ***lineItems*** містить вбудований масив документів і відповідає таблиці ***Товари_накладних*** в реляційної моделі.

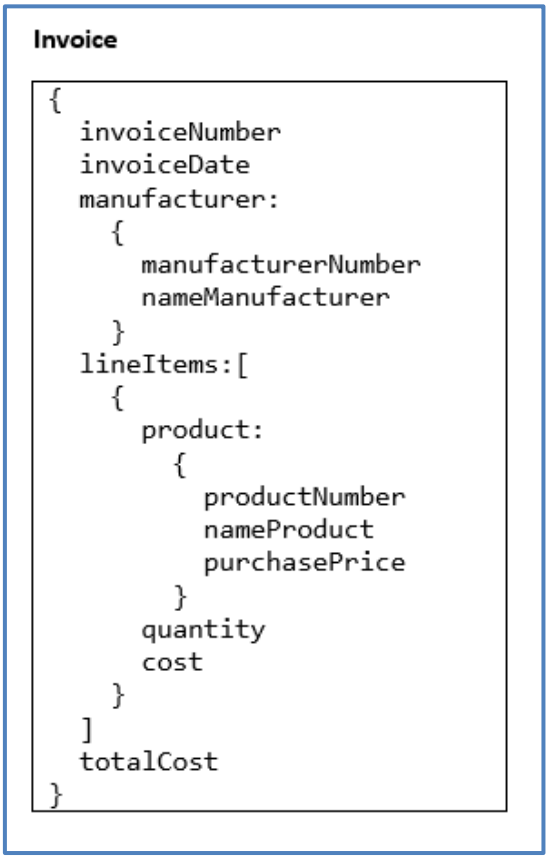


Рис. 7.14. **Схема документа *Накладна* у форматі квазі-JSON**

7.4. Створення бази даних і колекцій документів в ній. Додавання даних

Завдання 4

Ознайомтеся з основними елементами бази даних:

установіть поточну базу даних;

створіть базу даних і колекції документів шляхом додавання нового документа;

вивезіть вміст усієї колекції.

Виконання

Для виконання завдання необхідно дотримуватись такої послідовності дій.

1. Як поточну встановіть базу даних ***market*** (рис. 7.15) командою

```
use market
```

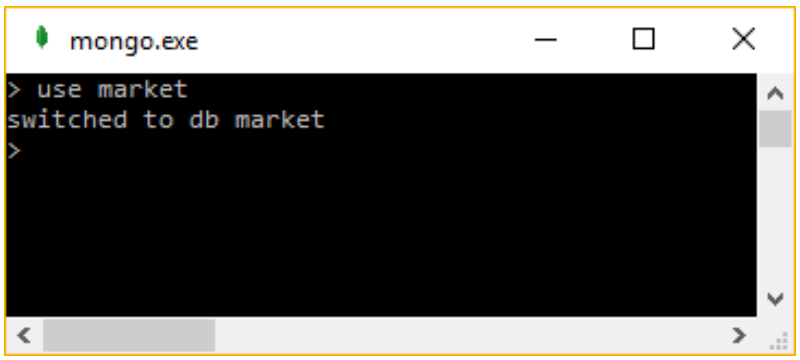


Рис. 7.15. **Установлення нової поточної бази даних**

Цієї бази ще немає. Переконайтеся в цьому. Вона з'явиться, коли буде записаний хоча б один документ.

Примітка: оскільки в подальшому команди будуть займати кілька рядків, для зручного їх набору спочатку треба формувати їх у програмі **Блокнот**, а потім скопіювати та вставити в командний рядок. В оболонці **mongo** вставка даних з буфера здійснюється клацанням правої клавіші миші.

2. Вставте дані з першої накладної в колекцію ***invoices***.

2.1. Введіть у **Блокнот** текст команди (рис. 7.16)

```
db.invoices.insert(
{
  invoiceNumber: '101',
  invoiceDate: new ISODate("2021-09-01"),
  manufacturer:
    {
      manufacturerNumber: 1,
      nameManufacturer: 'Х/з "Салтівський"',
    },
  lineItems:[
    {
      product:
        {
          productNumber: 1,
          nameProduct: 'Хліб "Український"',
          purchasePrice: 10
        },
      quantity: 200,
      cost: 2000
    },
```

```
    {
      product:
        {
          productNumber: 2,
          nameProduct: 'Батон "Молочний"',
          purchasePrice: 9.10
        },
      quantity: 210,
      cost: 1911
    }
  ],
  totalCost: 3911 }
)
```

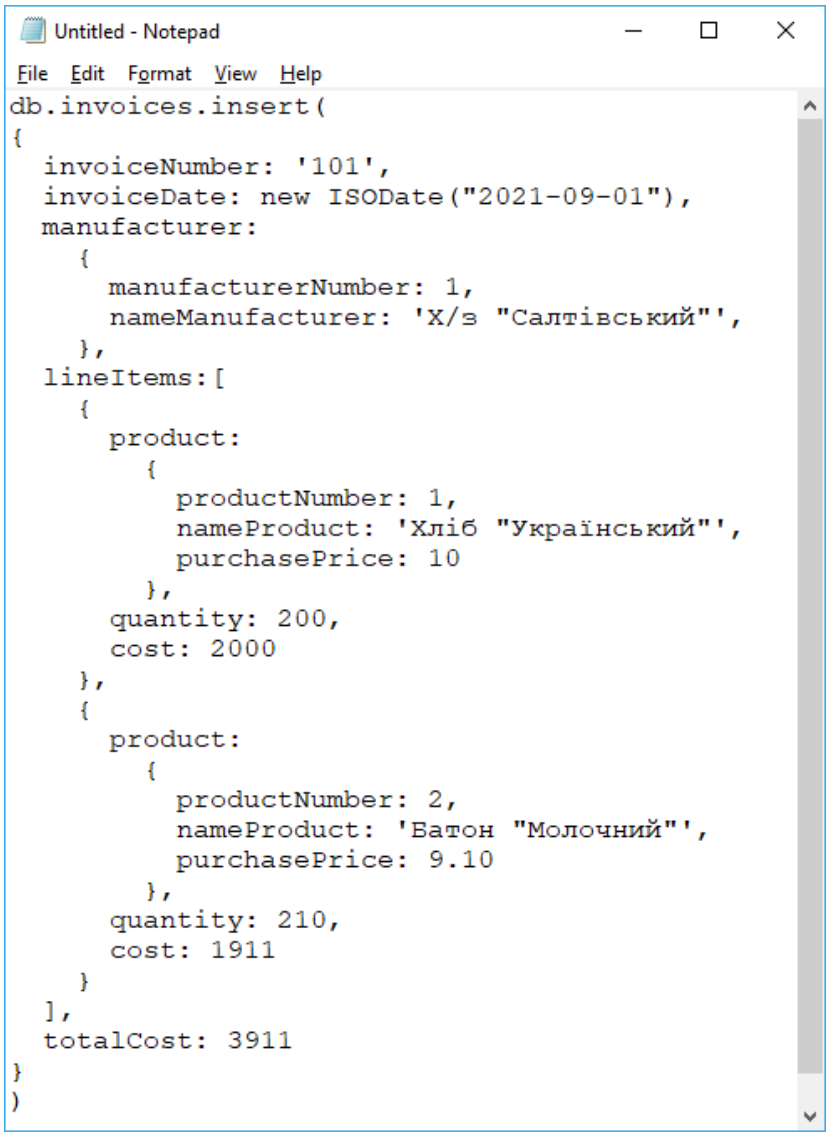


Рис. 7.16. **Текст команди у Блокноті**

2.2. Виділіть текст, що міститься в блокноті, скопіюйте його в буфер (Ctrl + C), потім перейдіть у вікно оболонки **mongo** та вставте вміст буфера в командний рядок, клацнувши в ній правою клавішею миші (рис. 7.17).

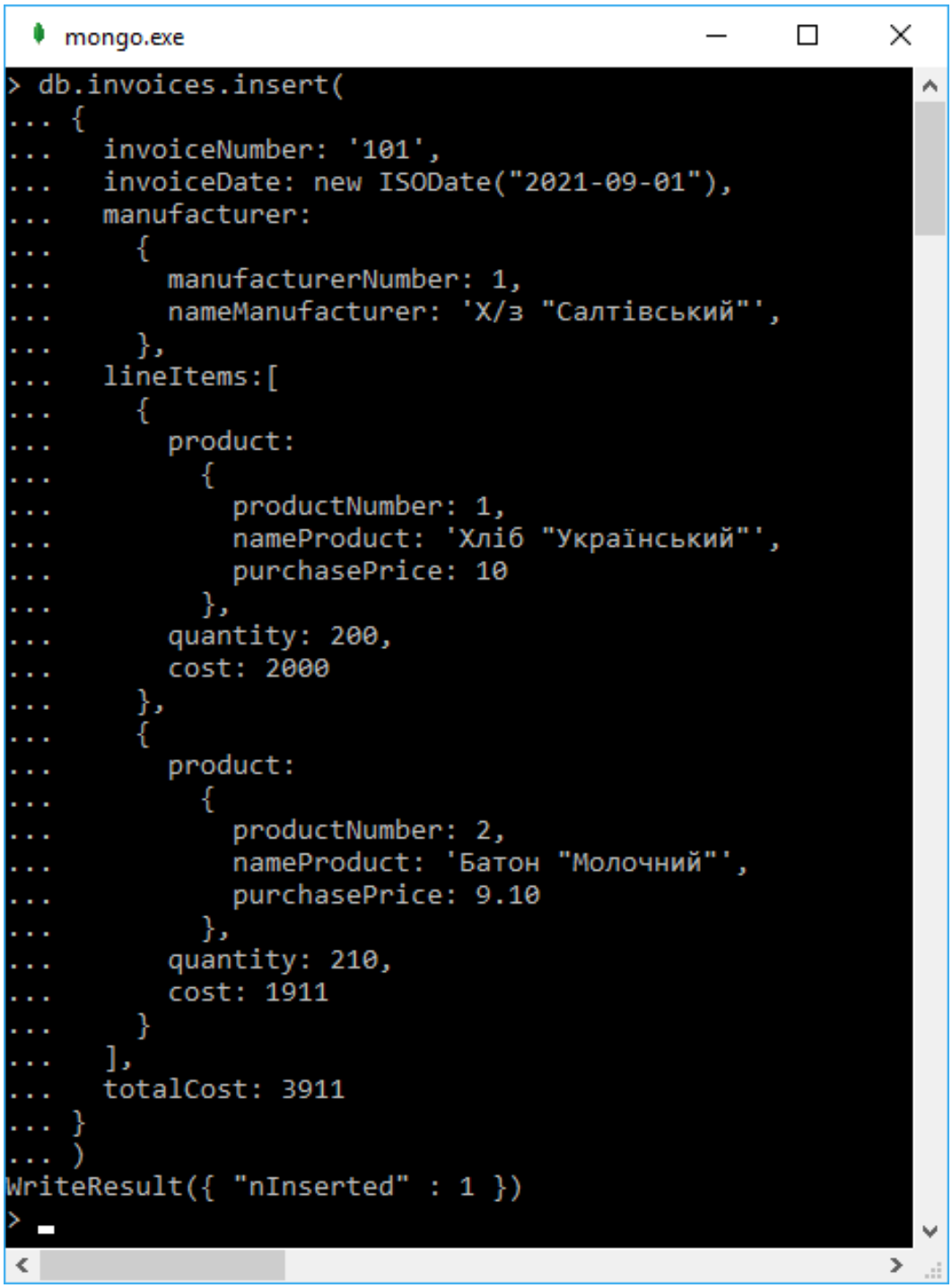


Рис. 7.17. **Вставлення даних з першої накладної**

2.3. Перегляньте вміст нової колекції ***invoices*** командою

```
db.invoices.find()
```

На рис. 7.18 показано результат виконання цієї команди.

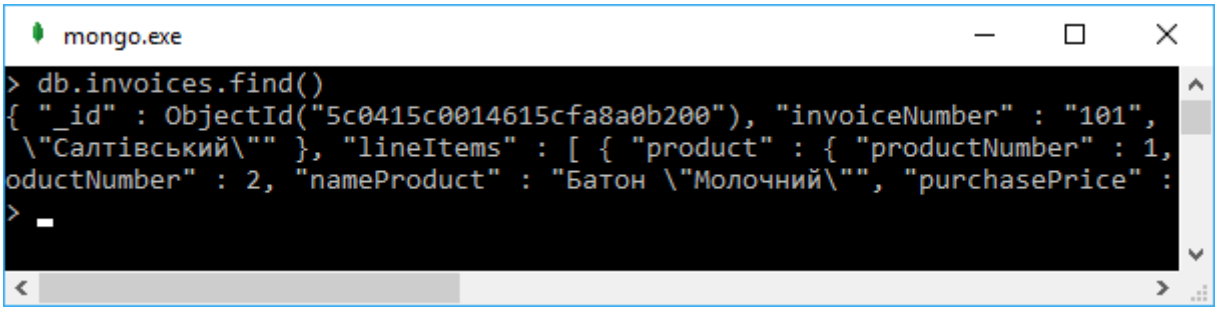


Рис. 7.18. **Вміст колекції *invoices***

Для подання колекції в JSON-форматі використовуйте метод ***pretty ()***. Виконайте команду

```
db.invoices.find().pretty()
```

На рис. 7.19 показано результат виконання цієї команди.

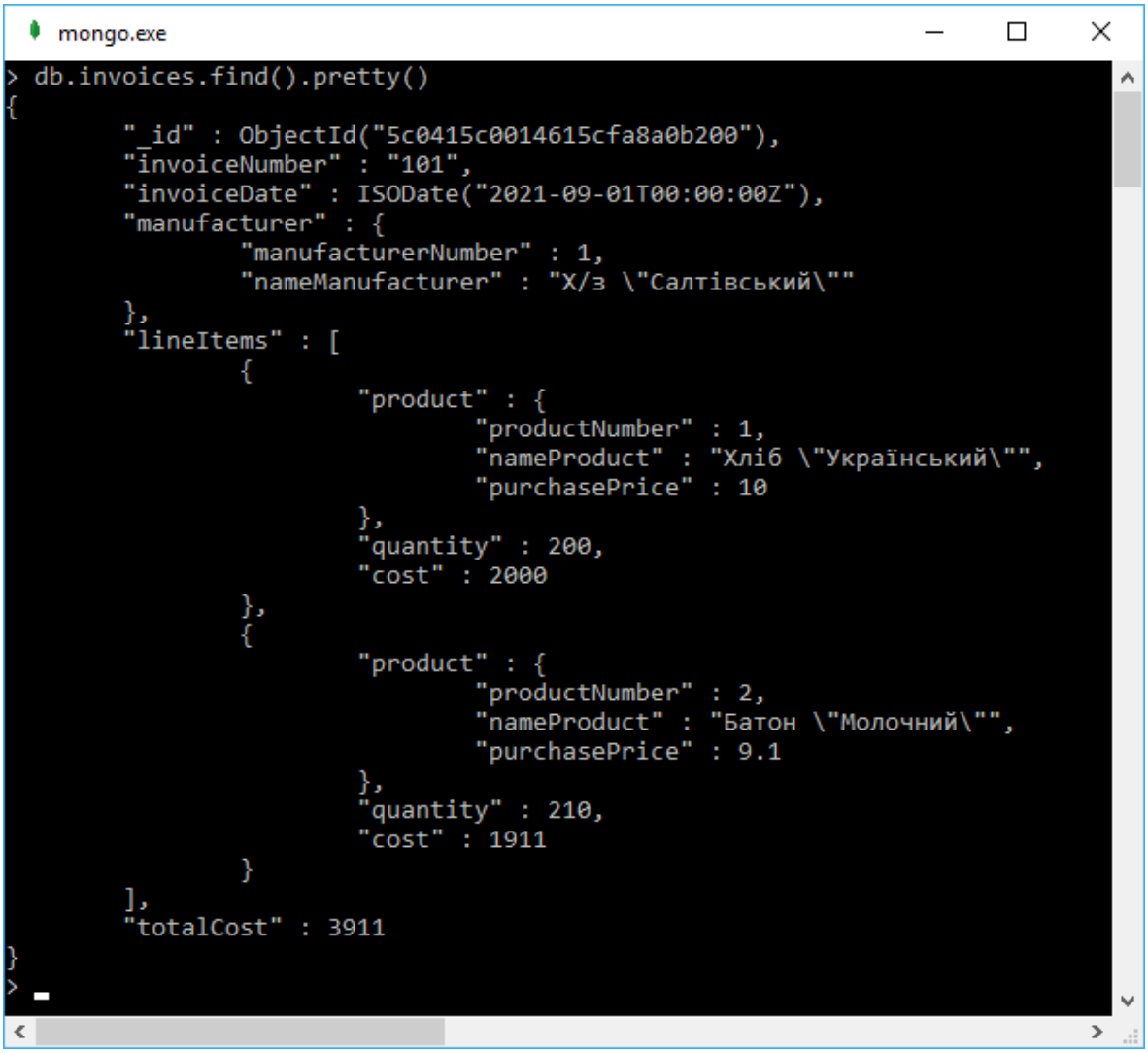


Рис. 7.19. **Колекції в JSON-форматі**

3. Вставте дані ще з двох накладних в колекцію ***invoices***.

3.1. Використовуючи **Блокнот**, введіть текст команди

```
db.invoices.insert([
{
  invoiceNumber: '201',
  invoiceDate: new ISODate("2021-09-01"),
  manufacturer:
    {
      manufacturerNumber: 1,
      nameManufacturer: 'Х/з "Кулиничі"',
    },
  lineItems:[
    {
      product:
        {
          productNumber: 1,
          nameProduct: 'Хліб "Український"',
          purchasePrice: 10
        },
      quantity: 210,
      cost: 2100
    },
    {
      product:
        {
          productNumber: 2,
          nameProduct: 'Батон "Молочний"',
          purchasePrice: 9.10
        },
      quantity: 150,
      cost: 1911
    },
    {
      product:
        {
          productNumber: 3,
          nameProduct: 'Булка з маком',
          purchasePrice: 8.65
        },
```

```
      quantity: 100,
      cost: 865
    }

  ],
  totalCost: 4330
},
{
  invoiceNumber: '202',
  invoiceDate: new ISODate("2021-09-02"),
  manufacturer:
    {
      manufacturerNumber: 1,
      nameManufacturer: 'Х/з "Кулиничі"',
    },
  lineItems:[
    {
      product:
        {
          productNumber: 1,
          nameProduct: 'Хліб "Український"',
          purchasePrice: 10
        },
      quantity: 250,
      cost: 2500
    },
    {
      product:
        {
          productNumber: 2,
          nameProduct: 'Батон "Молочний"',
          purchasePrice: 9.10
        },
      quantity: 100,
      cost: 910
    },
    {
      product:
        {
          productNumber: 3,
```

```
        nameProduct: 'Булка з маком',
        purchasePrice: 8.65
       },
     quantity: 100,
     cost: 865
   }

 ],
 totalCost: 4275
}
])
```

На рис. 7.20 показано результат виконання цієї команди.

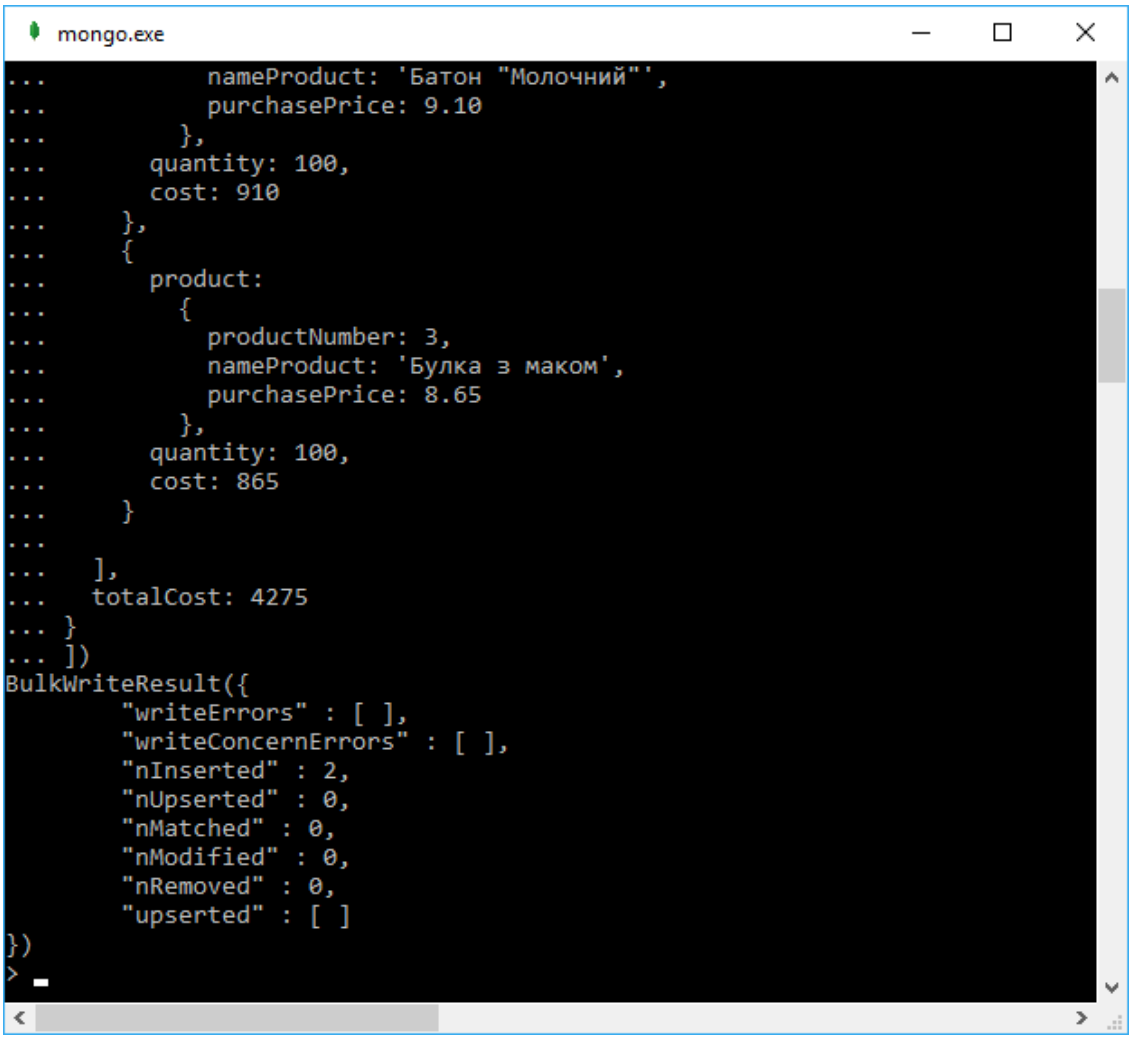


Рис. 7.20. **Вставлення даних ще за двома накладними в колекцію *invoices***

Примітка: у разі введення відразу декількох документів методом **insert** вони подаються як елементи масиву (беруться в квадратні дужки та розділяються комою).

У діагностиці після введення наводиться інформація про результат введення масиву (див. рис. 7.20).

3.2. Перегляньте вміст усієї колекції ***invoices*** командою

```
db.invoices.find().pretty()
```

На рис. 7.21 показано результат виконання цієї команди.

Рис. 7.21. **Виведення вмісту всієї колекції *invoices***

7.5. Оновлення даних

7.5.1. Заміна цілого документа

Завдання 5

Додати дані про поставку товару "Батон Слобожанський" у накладну з номером ***202***.

Ідея розв'язання

Оновлення даних в базі відбувається за таким алгоритмом.

1. Шукається за заданим номером накладна і її вміст зберігається в змінній ***doc***.

2. У змінну ***newItem*** записуються дані про поставку товару "Батон Слобожанський".

3. Додаються дані, що містяться в змінній ***newItem***, в масив ***lineItems*** документа ***doc*** як новий елемент.

4. Зберігається в базі даних змінений в пам'яті документ замість попереднього екземпляра цього документа.

Таким чином, додавання нового елемента в масив ***lineItems*** реалізується як оновлення документа ***invoice***. Оскільки при додаванні даних в існуючий документ його розмір збільшується, дані про всю накладну автоматично переміщаються в нове місце диска.

Виконання

Для виконання завдання необхідно дотримуватись такої послідовності дій.

1. Використовуючи **Блокнот**, введіть текст команд для реалізації алгоритму додавання даних про поставку товару "Батон Слобожанський" у накладну з номером ***202***.

```
var doc=db.invoices.findOne({invoiceNumber: '202'});
var newItem={
    product:
     {
       productNumber: 3,
       nameProduct: 'Батон "Слобожанський"',
       purchasePrice: 9
     },
```

```
    quantity: 200,
    cost: 1800
  };
doc.lineItems.push(newItem);
db.invoices.update({invoiceNumber: '202'}, doc);
```

На рис. 7.22 показано результат виконання цієї команди.

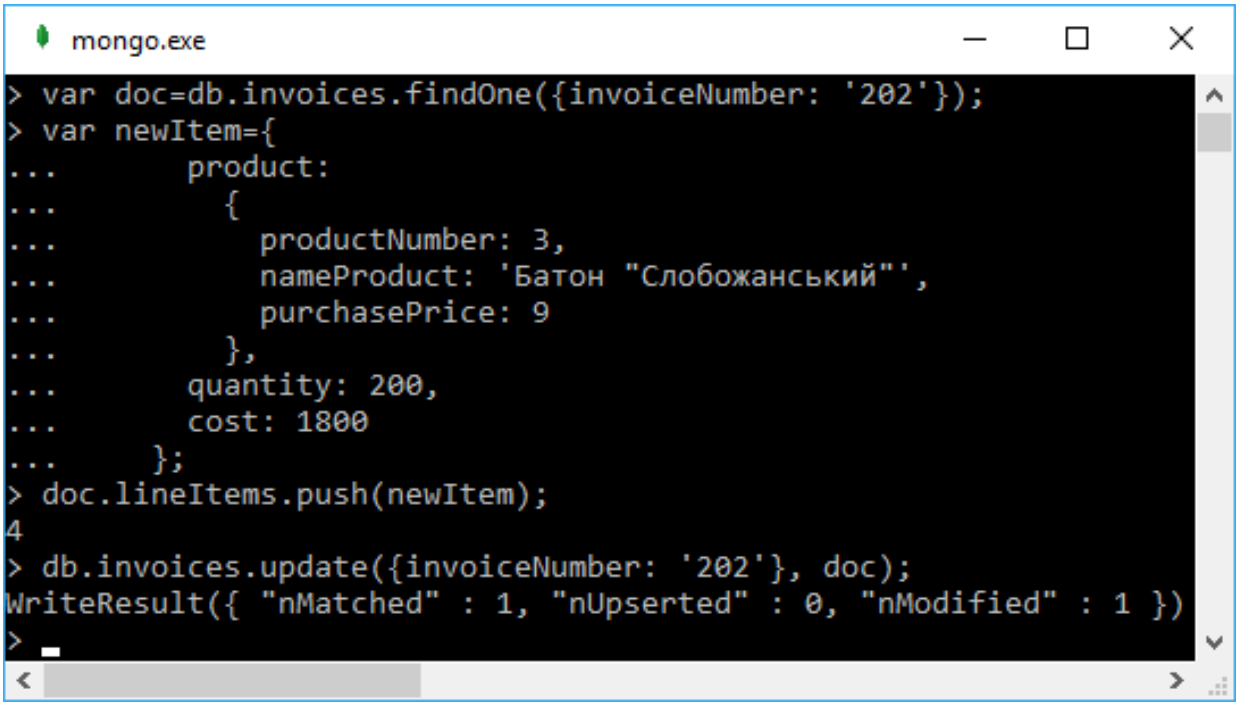


Рис. 7.22. **Додавання даних про поставку товару "Батон Слобожанський" у накладну з номером *202***

Примітка: метод ***push()*** додає вказаний елемент у масив.

2. Перегляньте вміст всієї колекції ***invoices*** командою:

```
db.invoices.find().pretty()
```

На рис. 7.23 показано дані зміненої накладної як результат виконання цієї команди. У накладній міститься вже чотири рядки з товарами. Але загальна вартість усієї накладної (***totalCost***) подана неправильно. Вона залишилася колишньою і виправляється певною операцією – спрямованою зміною (зміною на місці).

Рис. 7.23. **Дані зміненої накладної після додавання інформації про новий товар**

7.5.2. Спрямована зміна

Завдання 6

Збільшити загальну вартість усієї накладної з номером ***202*** на вартість доданого товару ***1 800 грн***.

Ідея розв'язання

Оновлення даних в базі відбувається за таким алгоритмом.

1. Шукається за заданим номером накладна.

2. Збільшується на ***1 800*** значення поля ***totalCost*** в знайденій накладній.

Отже, зміна значення поля ***totalCost*** в документі здійснюється на місці (в базі даних), тому додаткові операції збереження в базі не потрібні.

Виконання

Для виконання завдання необхідно дотримуватись такої послідовності дій.

1. Введіть текст команди, яка реалізує відразу два кроки алгоритму зміни значення поля ***totalCost*** у накладній з номером ***202***.

```
db.invoices.update({invoiceNumber: '202'}, {$inc: {totalCost: 1800}})
```

На рис. 7.24 показано результат виконання цієї команди.

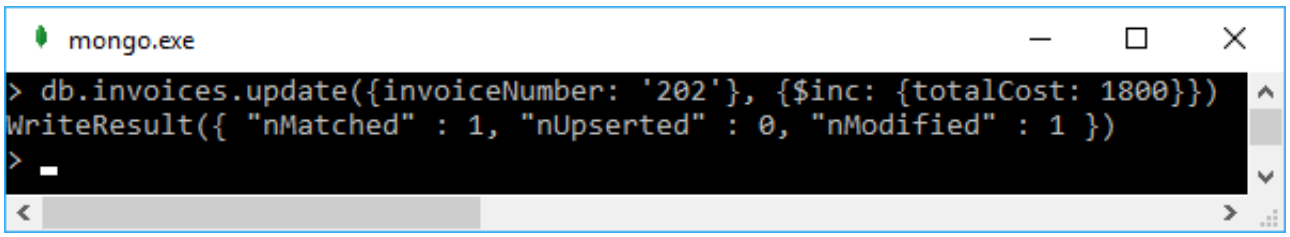


Рис. 7.24. **Зміна значення поля *totalCost* у накладній з номером *202***

Примітка: модифікатор ***$inc*** збільшує зазначений елемент на задану величину.

2. Перегляньте вміст усієї колекції ***invoices*** командою

```
db.invoices.find().pretty()
```

На рис. 7.25 показано дані зміненої накладної як результат виконання цієї команди.

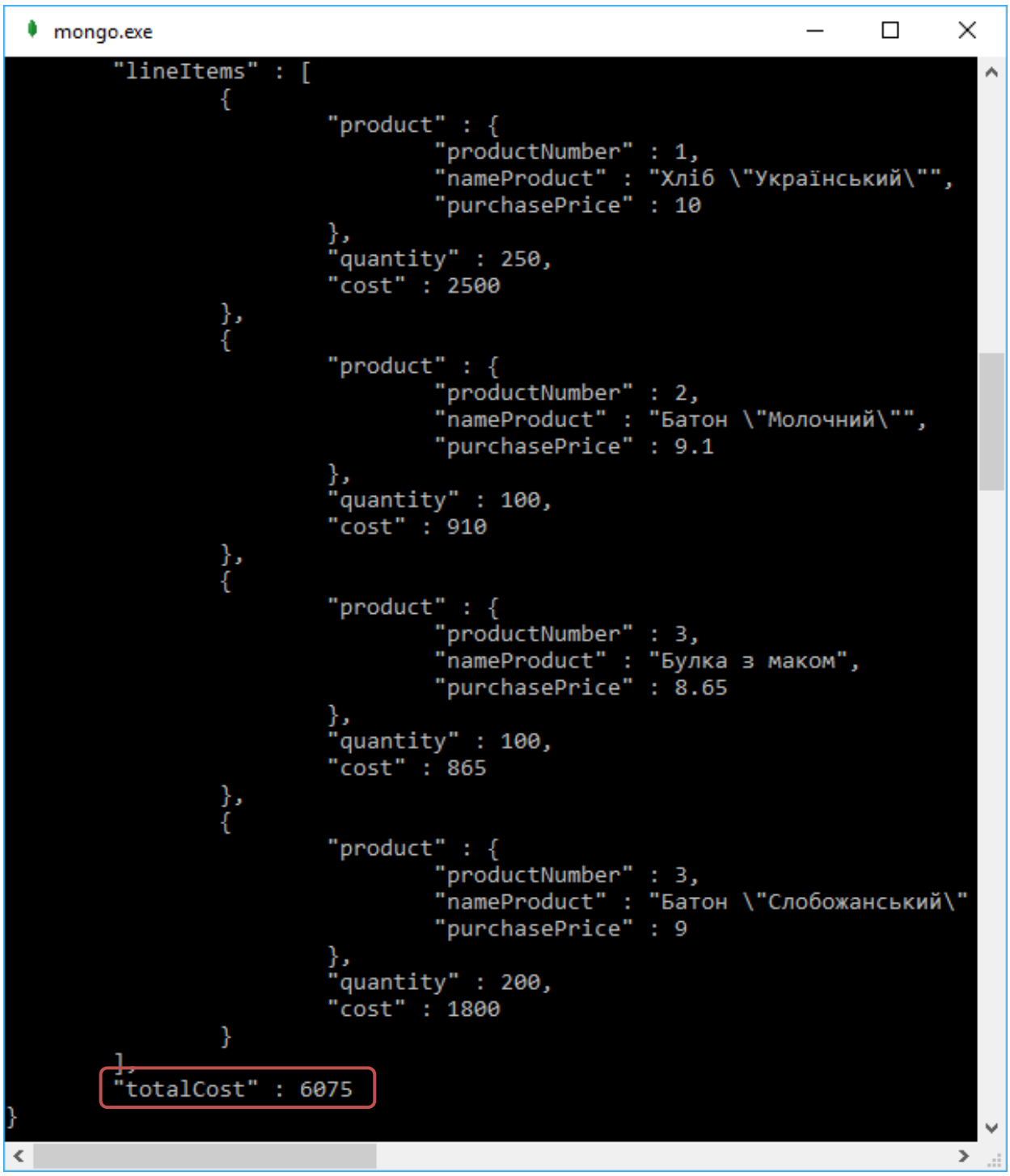


Рис. 7.25. **Збільшене значення загальної вартості усієї накладної з номером *202***

7.5.3. Оновлення/вставка

Завдання 7

Додати в колекцію ***invoices*** дані про постачання першого виробника 02.09.2021 за накладною з номером ***102***, якщо вони були такими самими, як і за його накладною з номером ***101***.

Ідея розв'язання

Для дублювання документів часто використовують третій параметр методу ***update()***. Йому встановлюють значення ***true***. У цьому разі зі спробою змінити заданий документ його спочатку шукають у колекції за заданим фільтром. Якщо знаходять, то замінюють знайдений документ новим, якщо ж ні – додають заданий документ у колекцію.

Додавання документа в базу шляхом відновлення існуючого відбувається за таким алгоритмом.

1. Шукається за заданим номером ***101*** накладна і її вміст зберігається у змінній ***doc***.

2. Видаляється ключове поле ***_id*** з документа, що зберігається в змінній ***doc***, щоб у процесі збереження його в базі даних сервер дав нове значення новому документу.

3. Встановлюється новий номер ***102*** накладної в змінній ***doc***.

4. Проводиться спроба замінити накладну, що зберігається в базі даних, на накладну, яка міститься в змінній ***doc*** з новим номером ***102***. Оскільки такої накладної ще немає в базі даних, вона зберігається як новий документ.

5. Оскільки в новому документі міститься дата старої накладної, вона змінюється на місці.

Примітка: крок 5 можна було об'єднати з кроком 4. Наявність кроку 5 пов'язано з бажанням показати дію модифікатора ***$set***, який встановлює нове значення поля на місці.

Виконання

Для виконання завдання необхідно дотримуватись такої послідовності дій.

1. Використовуючи **Блокнот**, введіть текст команд для реалізації кроків 1 – 4 алгоритму додавання документа в базу шляхом поновлення існуючого.

```
var doc=db.invoices.findOne({invoiceNumber: '101'});
delete doc["_id"];
doc.invoiceNumber='102';
db.invoices.update({invoiceNumber: '102'}, doc, true);
```

На рис. 7.26 показані результати виконання цих команд.

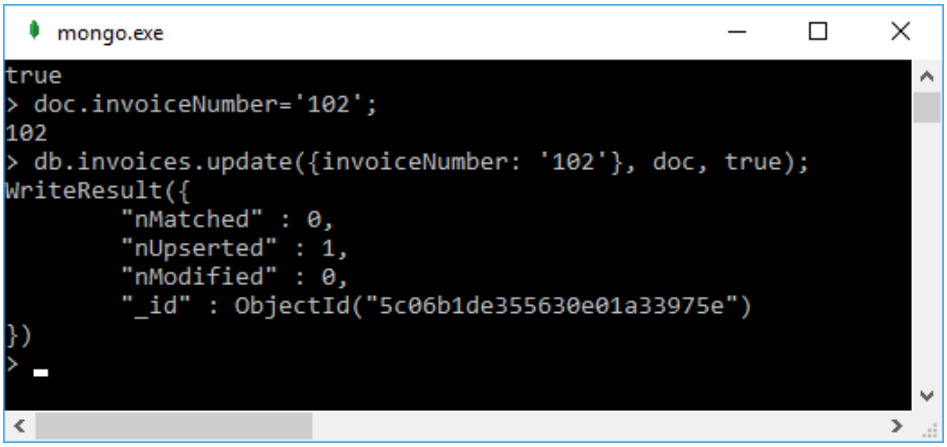


Рис. 7.26. **Результати виконання команд, що реалізують кроки 1 – 4 алгоритму**

2. Введіть текст команди для реалізації кроку 5 алгоритму (зміна дати на місці).

```
db.invoices.update({invoiceNumber: '102'}, {$set: {invoiceDate: new ISODate("2021-09-02") }})
```

На рис. 7.27 показано результат виконання цієї команди.

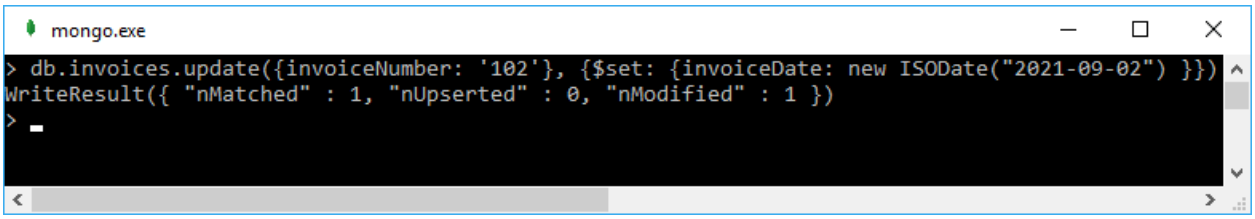


Рис. 7.27. **Результати виконання команди, що реалізує зміну дати на місці**

3. Перевірте кількість документів у колекції ***invoices*** (раніше їх було три). Для цього виконайте таку команду:

```
db.invoices.count()
```

На рис. 7.28 показано результат виконання цієї команди.

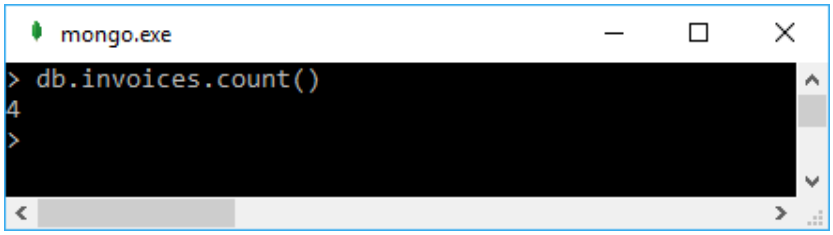


Рис. 7.28. **Кількість документів у колекції**

4. Перегляньте вміст нового документа (накладна з номером ***102***) командою

```
db.invoices.find({invoiceNumber: '102'}).pretty()
```

На рис. 7.29 показані дані нової накладної.

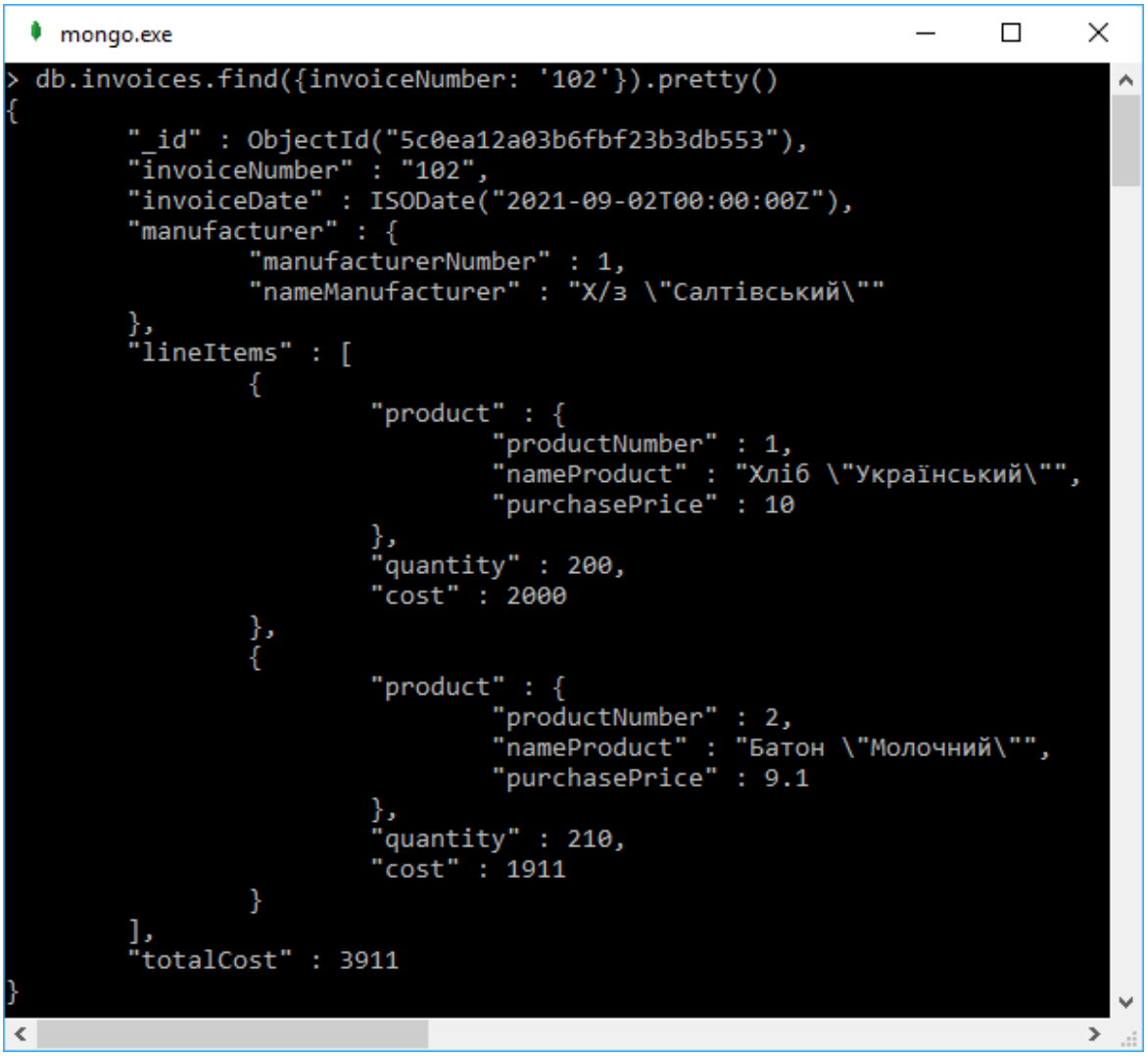


Рис. 7.29. **Дані накладної з номером *102***

Усі дані продубльовано правильно, змінені номер накладної та дата, згенеровано нове значення ключа.

7.6. Видалення даних

У цьому розділі показано, як проводиться видалення різних елементів з накладної з номером ***102***. Проте не слід піклуватися про цілісність даних, оскільки в кінці розділу видаляється вся накладна.

7.6.1. Видалення елемента з масиву

Завдання 8

Видалити дані про товар з номером 2 з накладної з номером ***102***.

Ідея розв'язання

Дані про товар подаються як елемент масиву ***lineItems***. Для видалення елемента масиву використовується оператор ***$pull***. Він видаляє кожне входження елемента в масив.

Цей оператор має такий формат:

```
{ $pull: { <field1>: <value|condition>, <field2>: <value|condition>, ... }
```

Оскільки видаляється лише елемент масиву вкладених документів, а сам документ залишається в базі даних (в усіченому вигляді), до всього документа застосовується операція оновлення (метод ***update()***).

Виконання

Для виконання завдання необхідно дотримуватись такої послідовності дій.

1. Введіть текст команди, яка реалізує операцію оновлення документа з видаленням елемента масиву.

```
db.invoices.update({invoiceNumber: '102'}, {$pull: {lineItems:
{"product.productNumber": 2}}})
```

На рис. 7.30 показано результат виконання цієї команди.

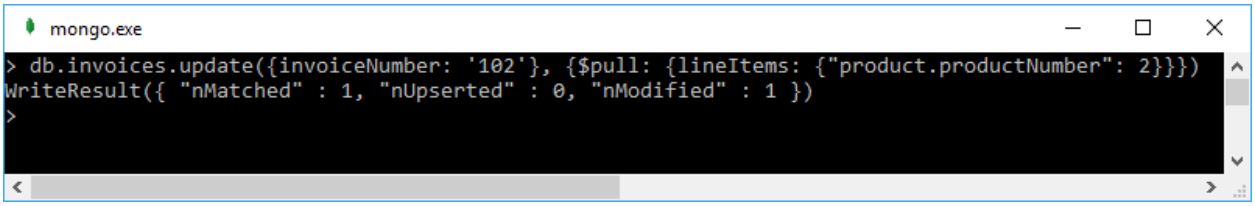


Рис. 7.30. **Видалення даних про товар з номером 2 з накладної з номером *102***

Примітка: номер товару елемента масиву, що видаляється, є елементом вкладеного документа ***product***, тому це поле має складене ім'я в операторі ***$pull***.

2. Перегляньте вміст накладної з номером ***102*** командою

```
db.invoices.find({invoiceNumber: '102'}).pretty()
```

На рис. 7.31 показано дані оновленої накладної. Зі списку товарів видалено дані про товар з номером ***2***. У ньому залишилися дані тільки про один товар.

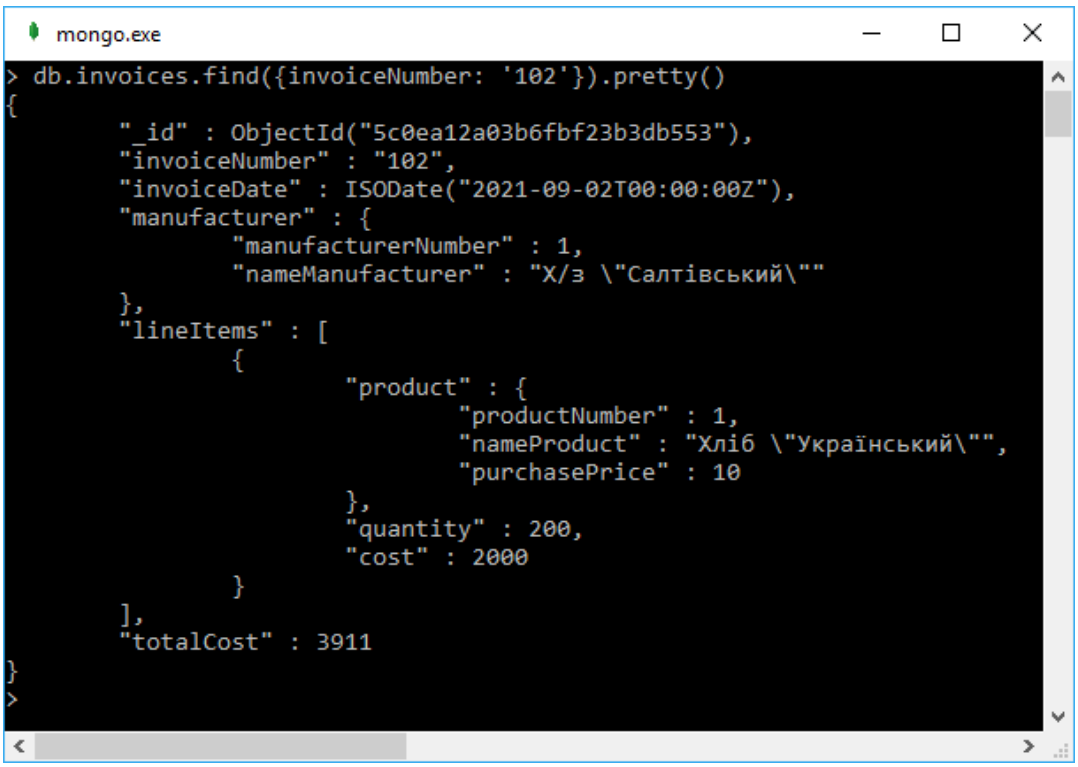


Рис. 7.31. **Дані оновленої накладної з номером *102***

7.6.2. Видалення поля з документа

Завдання 9

Видаліть дані про виробника з накладної з номером ***102***.

Ідея розв'язання

Оскільки видаляється лише елемент документа, а сам документ залишається в базі даних (в усіченому вигляді), до всього документа застосовується операція оновлення (метод ***update()***).

Для видалення поля з документа використовується оператор ***$unset***. Він має такий вигляд

```
{ $unset: { <field1>: "", ... } }
```

Виконання

Для виконання завдання необхідно дотримуватись такої послідовності дій.

1. Введіть текст команди, яка реалізує операцію оновлення документа з видаленням поля.

```
db.invoices.update({invoiceNumber:'102'},{$unset:{manufacturer:""}})
```

На рис. 7.32 показано результат виконання цієї команди.

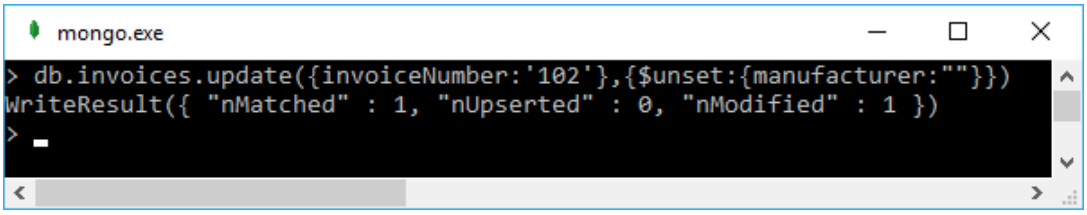


Рис. 7.32. **Видалення даних про виробника з накладної з номером *102***

2. Перегляньте вміст накладної з номером ***102*** командою:

```
db.invoices.find({invoiceNumber: '102'}).pretty()
```

На рис. 7.33 показано дані оновленої накладної. У накладній з номером ***102*** відсутній вкладений документ ***manufacturer***.

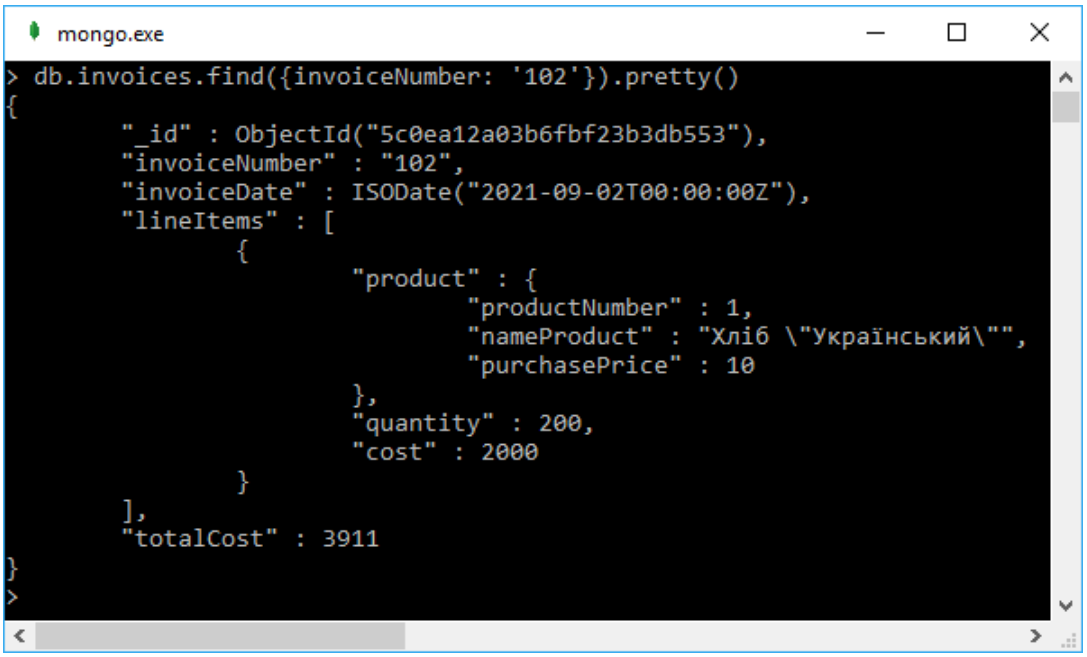


Рис. 7.33. **Дані за накладною з номером *102* після видалення відомостей про виробника**

7.6.3. Видалення документа з колекції

Завдання 10

Видаліть дані про всі накладні з номером ***102***.

Ідея розв'язання

Оскільки видаляється весь документ, то застосовується операція видалення (метод ***remove()***), яка має такий вигляд

```
db.collection.remove(<query>, <justOne>)
```

Тут

<query> – умова, якій задовільнюють документи, що видаляються;

<justOne> – видаляється лише перший знайдений документ (***true***) або видаляються усі документи, що задовільнюють умові (***false***).

Виконання

Для виконання завдання необхідно дотримуватись такої послідовності дій.

1. Введіть текст команди, яка реалізує операцію видалення документа з колекції.

```
db.invoices.remove({invoiceNumber: '102'}, true)
```

На рис. 7.34 показано результат виконання цієї команди.

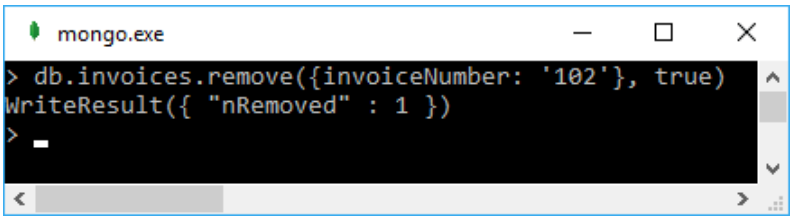


Рис. 7.34. **Видалення документа з колекції**

2. Перевірте кількість документів у колекції ***invoices*** (раніше їх було чотири). Для цього виконайте таку команду

```
db.invoices.count()
```

На рис. 7.35 показано результат виконання цієї команди.

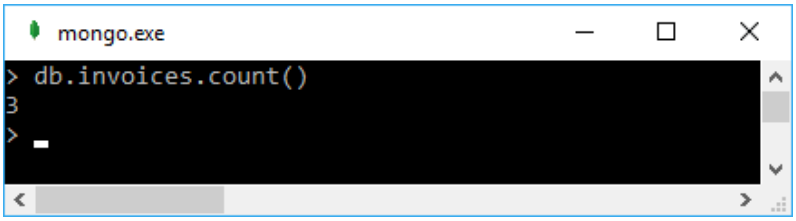


Рис. 7.35. **Кількість документів у колекції**

3. Перегляньте вміст усієї колекції ***invoices*** командою:

```
db.invoices.find().pretty()
```

На рис. 7.36 показано дані останньої (третьої) накладної. Вони збігаються з тими, які були раніше. Операція видалення документа з колекції проведена коректно.

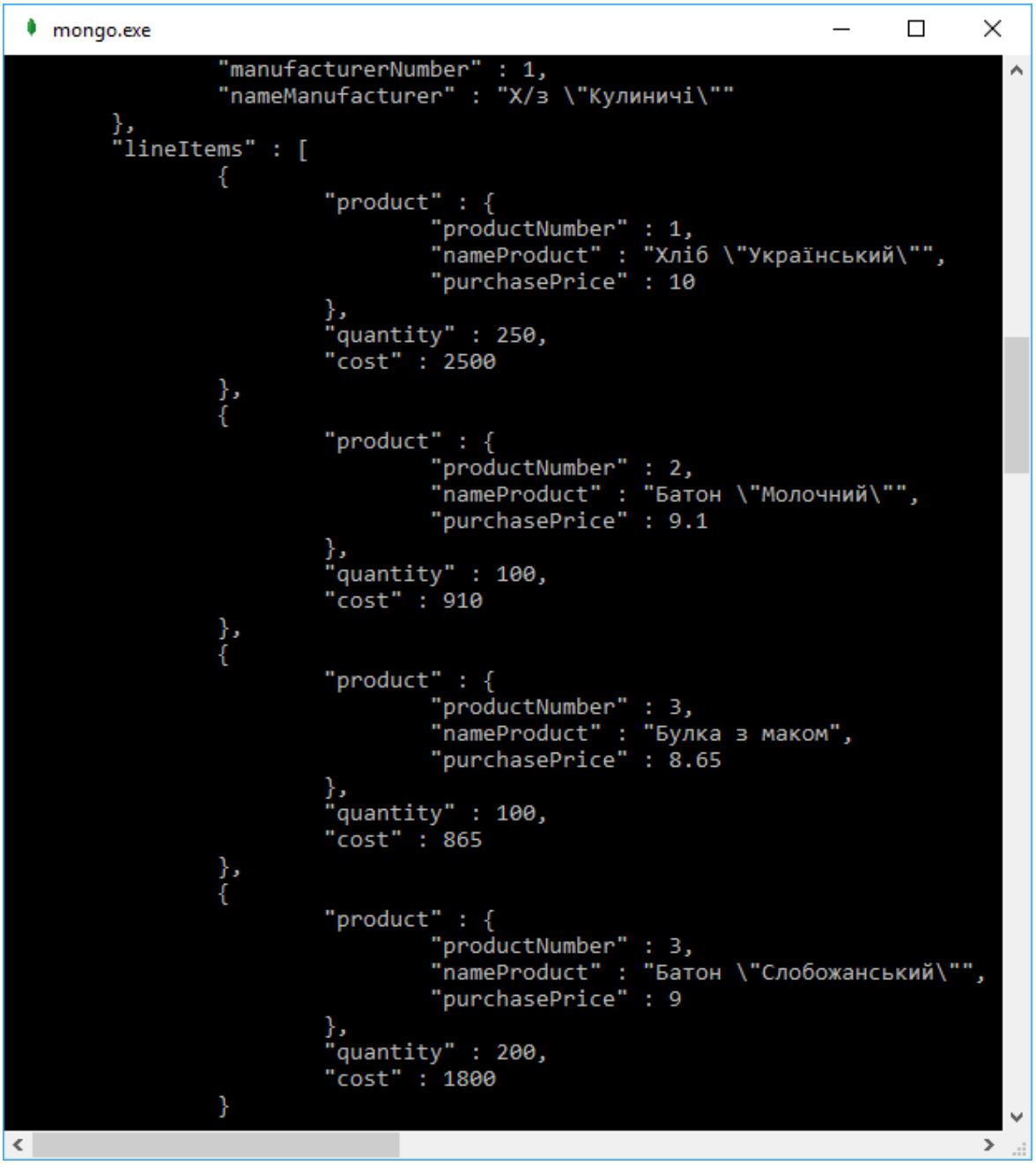


Рис. 7.36. **Дані останньої накладної після видалення накладної з номером *102***

7.7. Вибірка даних з бази

7.7.1. Одинарна умова

Завдання 11

Виведіть всі накладні, в яких є товар "Булка з маком".

Ідея розв'язання

Назва товару ***nameProduct*** є полем вкладеного документа ***product***, який, у свою чергу, є елементом масиву ***lineItems***. Отже, для задавання

умови пошуку потрібно використовувати складене ім'я, яке обов'язково забрати в лапки. Тому умова фільтрації має такий вигляд:

```
"lineItems.product.nameProduct":"Булка з маком"
```

Виконання

Введіть текст команди, яка реалізує операцію пошуку документів у колекції.

```
db.invoices.find({"lineItems.product.nameProduct":"Булка з маком"}).pretty()
```

На рис. 7.37 показано результат виконання цієї команди. Знайдено дві накладні з товаром "Булка з маком".

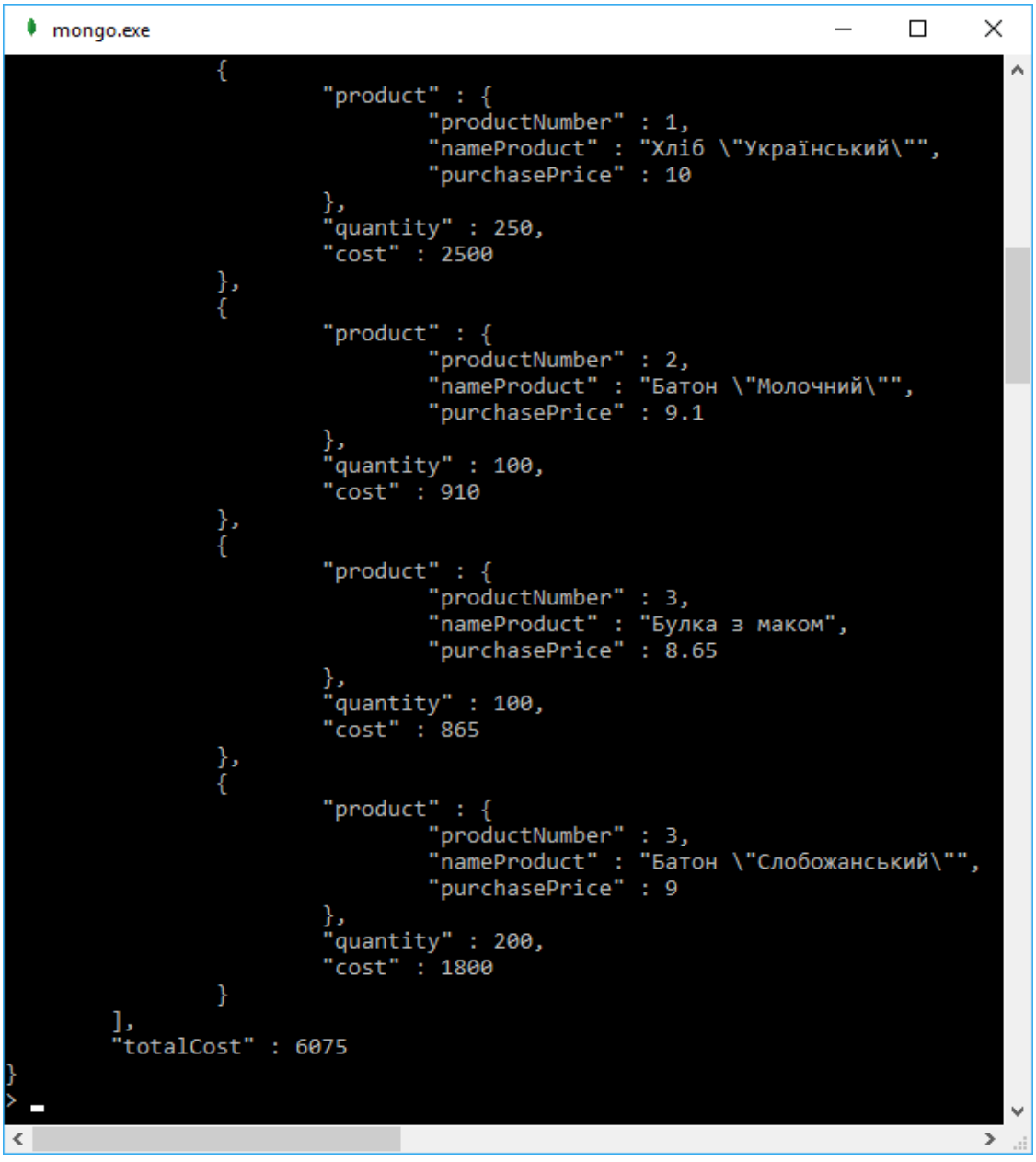


Рис. 7.37. **Накладні, в яких є товар "Булка з маком"**

7.7.2. Складена умова

Завдання 12

Виведіть всі накладні, в яких є товар "Булка з маком" і загальна вартість менша 5 000 грн.

Ідея розв'язання

Дві та більше умов перераховуються через кому в першому параметрі методу ***find()***. Передбачається, що між ними вказано логічну операцію ***And***.

Виконання

Введіть текст команди, яка реалізує операцію пошуку документів у колекції

```
db.invoices.find({"lineItems.product.nameProduct":"Булка з маком",
totalCost: { $lt: 5000}}).pretty()
```

На рис. 7.38 показано результат виконання цієї команди. Знайдено тільки одну накладну, яка задовільнює усім зазначеним умовам.

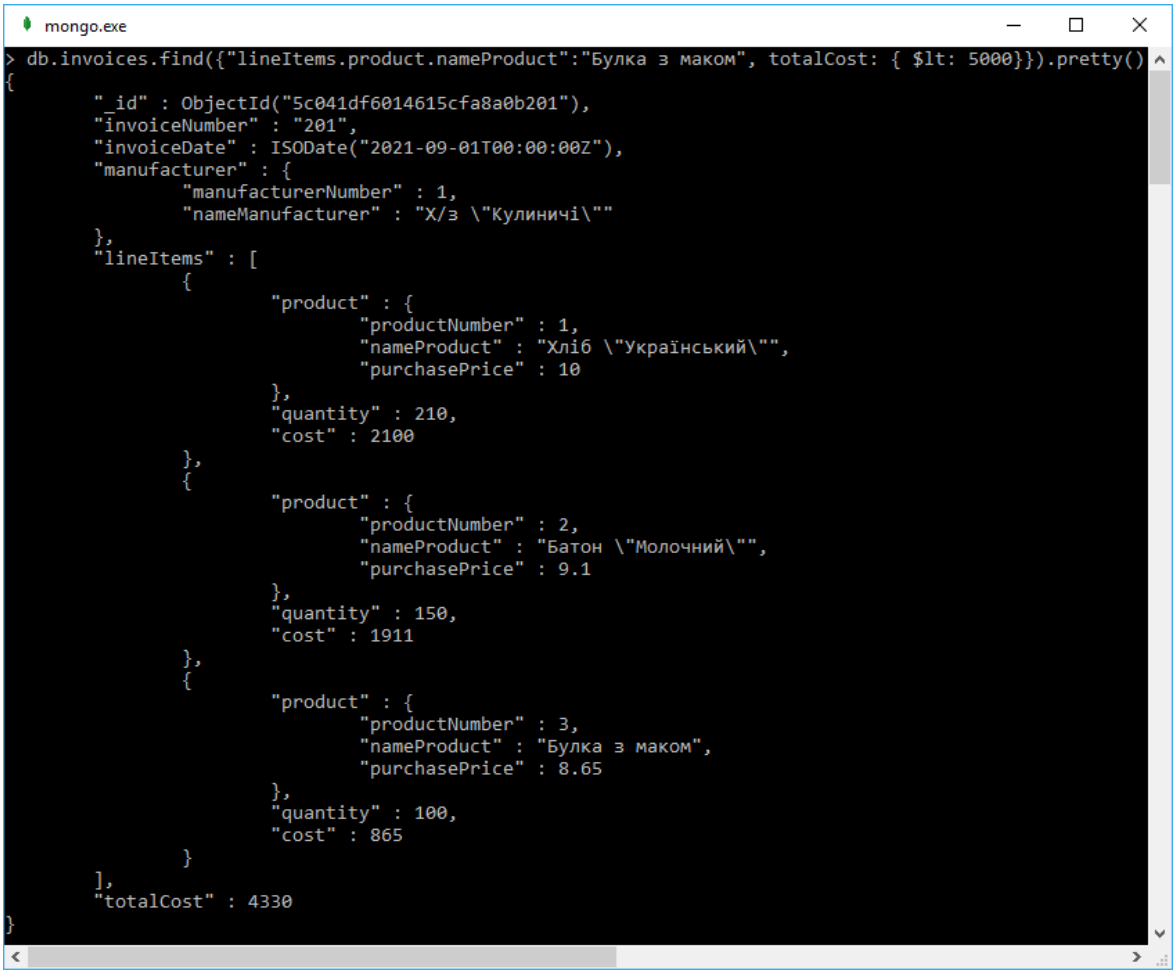


Рис. 7.38. **Накладна, яка задовільнює складеній умові**

7.7.3. Проекція

Завдання 13

Виведіть номер накладної, дату та загальну вартість за накладними, в яких загальна вартість менша 5 000 грн, проте не треба виводити поле ***_id*** документа.

Ідея розв'язання

За замовчуванням метод ***find()*** виводить усі поля відібраних документів. Щоб явно вказати, які поля потрібно виводити, а які – ні, перераховують їхні імена в другому параметрі методу ***find()*** і через двокрапку ставлять цифру ***1*** або ***0***. Цифра ***1*** означає, що поле виводиться, а ***0*** – ні. Якщо зазначено другий параметр (проекція), записують або поля, що виводяться, або навпаки.

Ключове поле ***_id*** за замовчуванням виводиться. Якщо його не потрібно виводити, задають зі значенням ***0***.

Виконання

Введіть текст команди, яка реалізує операцію виведення полів документів із заданими умовами.

```
db.invoices.find({totalCost: {$lt: 5000}},
{invoiceNumber:1,invoiceDate:1,totalCost:1,_id:0})
```

На рис. 7.39 показано результат виконання цієї команди. Знайдено дві накладні, які задовільнюють усім зазначеним умовам. Оскільки дані, що виводяться за кожним документом, поміщаються в одному рядку, метод ***pretty()*** не використовувався.

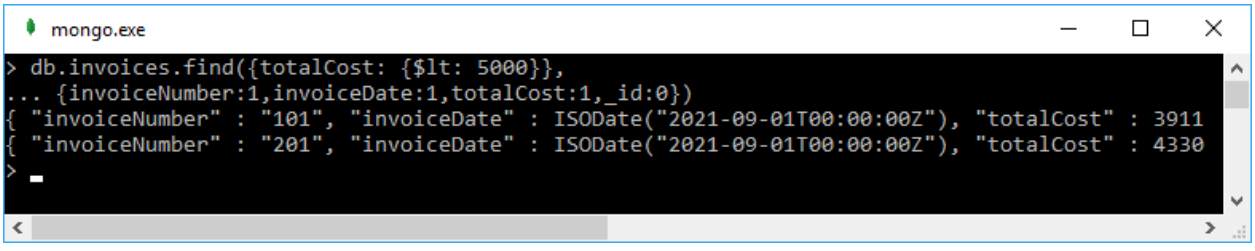


Рис. 7.39. **Накладні, які задовільнюють складеній умові**

7.7.4. Використання JavaScript

Завдання 14

Знайдіть накладні, в яких у назві виробника зустрічається текст ***Куликиничі***, а загальна вартість знаходиться між 3 000 і 5 000 грн.

Ідея розв'язання

MongoDB дозволяє створювати запити, використовуючи мову JavaScript. Зокрема, застосовуючи мову JavaScript, можна скласти як завгодно складну умову. Її вказують після оператора ***$where*** в першому параметрі методу ***find()***.

Виконання

Введіть текст команди, яка реалізує операцію пошуку документів із заданими умовами.

```
db.invoices.find({
  $where:
  "(this.manufacturer.nameManufacturer.indexOf('Кулиничі')!= -1)
&&(this.totalCost>=3000)&&(this.totalCost<=5000)"
  }).pretty()
```

На рис. 7.40 показано результат виконання цієї команди. Знайдено тільки одну накладну, яка задовільнює усім зазначеним умовам.

Рис. 7.40. **Накладна, яка задовільнює заданій умові**

7.7.5. Сортування

Завдання 15

Виведіть номери накладних, їх ***_id*** і загальну вартість у порядку спадання останньої.

Ідея розв'язання

Сортування задається як метод, який застосовується до відібраних даних. Його параметр містить пари *поле: значення*. Значенням може бути ***1*** (сортування за зростанням) або ***-1*** (сортування за спаданням). Указуючи кілька пар, задають багаторівневе сортування.

Виконання

Введіть текст команди, яка реалізує операцію виведення полів документів із заданими умовами.

```
db.invoices.find({},{invoiceNumber:1,totalCost:1}).sort({totalCost: -1})
```

На рис. 7.41 показано результат виконання цієї команди. Дані за накладними відсортовано в порядку спадання загальної вартості.

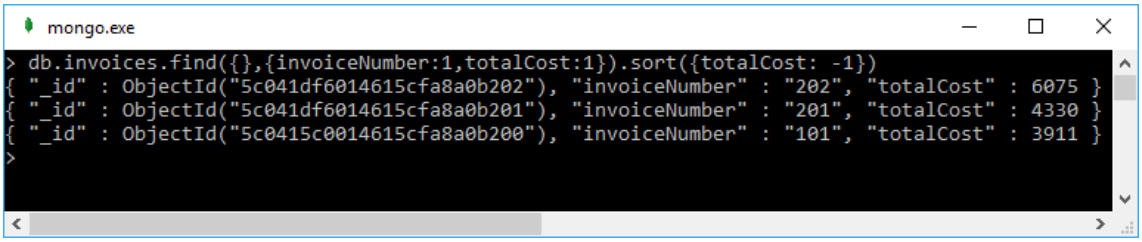


Рис. 7.41. **Відсортовані дані**

7.7.6. Пейджинг

Завдання 16

Нехай на кожну сторінку виводяться дані за однією накладною. Виведіть дані на третю сторінку (пропустити два документа та вивести один).

Ідея розв'язання

Пейджингом називають розбиття даних на сторінки. Процедура реалізується як підвибірка з вибраних даних і використовується для виведення даних частинами. Для організації пейджингу застосовують методи ***skip()*** – кількість пропущених документів і ***limit()*** – максимальна кількість виведених документів.

Виконання

Введіть текст команди, яка реалізує операцію виведення даних на третю сторінку.

```
db.invoices.find({}).skip(2).limit(1).pretty()
```

На рис. 7.42 показано результат виконання цієї команди. На третю сторінку виведено дані третьої накладної.

Рис. 7.42. **Дані третьої сторінки**

7.7.7. Курсори

Завдання 17

Визначте середню вартість накладних.

Ідея розв'язання

Метод **find()** повертає курсор, який є об'єктом, що дозволяє обробляти кожний відібраний документ або групу документів. Для виконання операцій з документами курсору використовують цикли, які реалізуються засобами мови JavaScript.

Для визначення середньої вартості накладних спочатку за допомогою циклу обчислюють суму вартості всіх накладних та їхню кількість, а потім ділять першу величину на другу.

Виконання

Введіть текст команд, які реалізують обчислення середньої вартості накладних.

```
var count = 0;
var sum = 0;
var avg = 0;
var cursor = db.invoices.find();
while (cursor.hasNext()) {
        doc = cursor.next();
        sum += doc.totalCost;
        count += 1
};
avg = sum / count;
```

На рис. 7.43 показано результат виконання цих команд. Середня вартість накладних становить 4 772 грн.

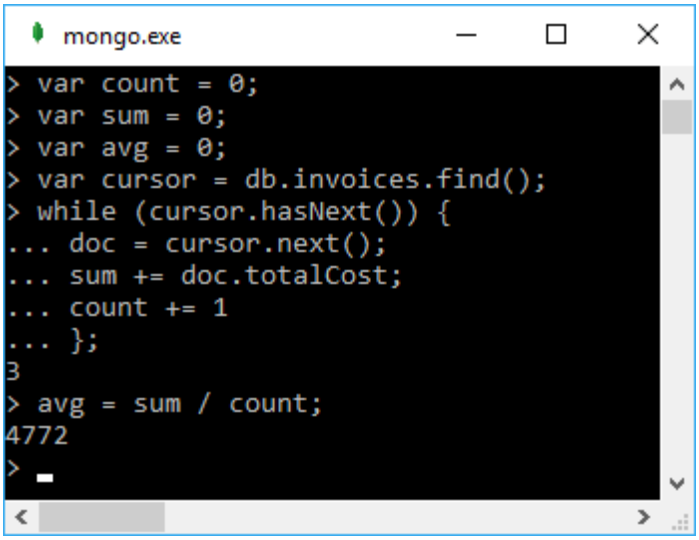


Рис. 7.43. **Обчислення середньої вартості накладних**

Завдання для самостійного виконання

1. Додати ще одну накладну від виробника ***Х/з "Кулиничі"***.

2. Замінити назву виробника на ***Х/З "Салтівський"*** в доданій у п. 1 накладній.

3. Додати ще один товар в останню накладну, відповідним чином змінивши загальну вартість товарів за цією накладною.

4. Вивести дати, назви товарів та їхню вартість за накладними від виробника ***Х/З "Салтівський"***.

5. Створити колекцію ***sales*** для зберігання даних про щоденні продажі в базі даних ***market***.

6. Вивести загальну вартість залишків товарів за весь період.

7. Створити базу даних за індивідуальним завданням і показати виконання CRUD-операцій у ній.

Лабораторна робота 8
Створення бази даних SQL Azure на порталі Microsoft Azure

Цілі роботи

1. Створення підписки для Microsoft Azure.

2. Підключення та розгортання БД засобами хмарної технології Microsoft Azure.

3. Міграція БД.

Перед виконанням роботи студент повинен знати:

основи проектування реляційних баз даних економічного характеру.

Після виконання лабораторної роботи студент повинен уміти:

зайти до порталу Microsoft Azure;

створювати нову базу даних і здійснювати її розгортання в Microsoft Azure;

створювати таблиці та запити для БД на порталі Microsoft Azure;

виконувати міграцію локальної БД на порталі Microsoft Azure.

Хід роботи

8.1. Створення підписки для роботи з порталом Microsoft Azure.

8.2. Створення нової бази даних і виконання її розгортання на порталі Microsoft Azure.

8.3. Створення таблиці за допомогою редактора БД порталу Microsoft Azure.

8.4. Виконання запитів до БД.

8.5. Виконання міграції локальної БД на портал Microsoft Azure.

Форма звітності

За результатами виконання роботи слід оформити електронний звіт засобами Word. В електронному звіті за кожним завданням подати текст завдання з коментарями згідно з наведеними нижче прикладами.

Критерії оцінювання

У 12-бальній системі оцінка результату захисту лабораторної роботи формується за такими правилами.

1. За завдання 8.1 – максимум 1 бал.

2. За завдання 8.2 – максимум 3 бали.

3. За кожне завдання 8.3 і 8.4 – максимум 2 бала.

4. За завдання 8.5 – максимум 4 бала.

5. За запізнення із захистом лабораторної роботи знімається 2 бала за кожний тиждень.

Отримана кількість балів з відповідей на кожне завдання лабораторної роботи підсумовується. У результаті такого підрахунку студент може отримати від 0 до 12 балів.

Рекомендована література: [14; 22].

Основні поняття

Microsoft Azure – це хмарний продукт Microsoft, який використовує мережу центрів обробки даних для створення, розгортання та управління службами та додатками в будь-якому місці. Він дозволяє додавати можливості в існуючу мережу через платформу в якості моделі сервісу (PaaS) і доручає Microsoft виконувати обчислювальні та мережеві операції за допомогою інфраструктури служби (IaaS).

Хмарна платформа Microsoft Windows Azure підтримує ряд мов програмування додатків, включаючи JavaScript, Python, .NET і Node.js. Інструменти цієї категорії також включають підтримку Visual Studio, комплектів розробки програмного забезпечення (SDK) і блокування.

База даних SQL Azure – це реляційна база даних, яка надається за моделлю "як послуга" (DBaaS), на основі останньої стабільної версії ядра СУБД Microsoft SQL Server. **База даних SQL** – це високопродуктивна, надійна та безпечна база даних, за допомогою якої можна створювати додатки і веб-сайти на основі даних, використовуючи будь-яку мову програмування, без необхідності управляти інфраструктурою.

Для використання бази даних SQL платформа Microsoft Azure надає доступ: до шаблонів швидкого запуску, прикладів і посібників.

Робота з SQL Azure достатньо проста, оскільки підтримується велика частина інструментів і засобів розробки, наявні у SQL Server.

За допомогою порталу Azure і SQL Server Management Studio можна виконувати такі дії: створювати базу даних на порталі Azure, налаштовувати правило брандмауера на рівні сервера на порталі Azure, підключатися до бази даних за допомогою SQL Server Management Studio, створювати таблиці за допомогою SSMS, запрошувати дані за допомогою SSMS та міграцію баз даних.

Важливо! Якщо у вас ще немає підписки Azure, створіть безкоштовний обліковий запис Azure, перш ніж починати роботу.

Практична частина

Попередні зауваження

У Microsoft Azure завжди існувала можливість розміщувати безкоштовні Azure Web Sites, однак для отримання доступу до хмари була необхідна реєстрація з кредитною карткою. У рамках програми Microsoft Imagine студенти можуть отримати доступ до Azure Web Sites і деяким іншим можливостям хмари Microsoft Azure шляхом простої реєстрації.

Хмара доступна для студентів безкоштовно і без кредитної картки (для реєстрації потрібно тільки e-mail і номер мобільного телефону).

Насамперед за все, необхідно пройти реєстрацію: завести (якщо у вас його немає) обліковий запис Microsoft (Microsoft account), верифікувати себе як студента на Microsoft Imagine, а потім пройти окремий процес реєстрації на Microsoft Azure.

Для доступу до всіх хмарних ресурсів слугує портал portal.azure.com, в який можна ввійти зі своїм Microsoft Account. Вигляд вікна Microsoft Account поданий на рис. 8.1.

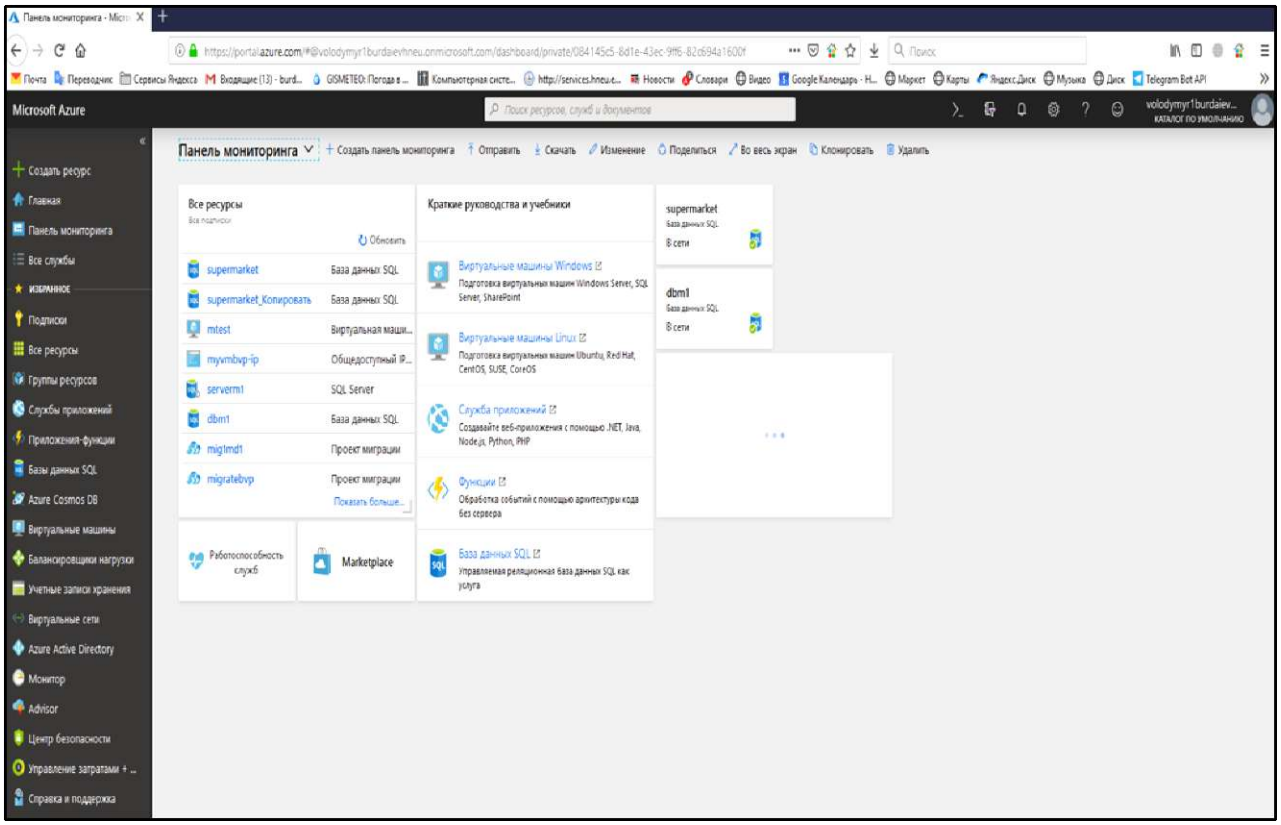


Рис. 8.1. **Вид вікна Microsoft Azure**

База даних Azure SQL створюється з певним набором обчислювальних ресурсів і ресурсів зберігання. База даних створюється в межах групи ресурсів Azure та логічного сервера бази даних SQL Azure.

8.1. Створення підписки для роботи з порталом Microsoft Azure

Завдання 1

Отримайте підписку для роботи з базою даних SQL на платформі Microsoft Azure.

Виконання

Для виконання завдання необхідно дотримуватись такої послідовності дій.

1. Створіть обліковий запис (акаунт) на сайті Microsoft live id https://account.microsoft.com.

2. Завантажте головну сторінку Microsoft Imagine https://imagine.microsoft.com (рис. 8.2).

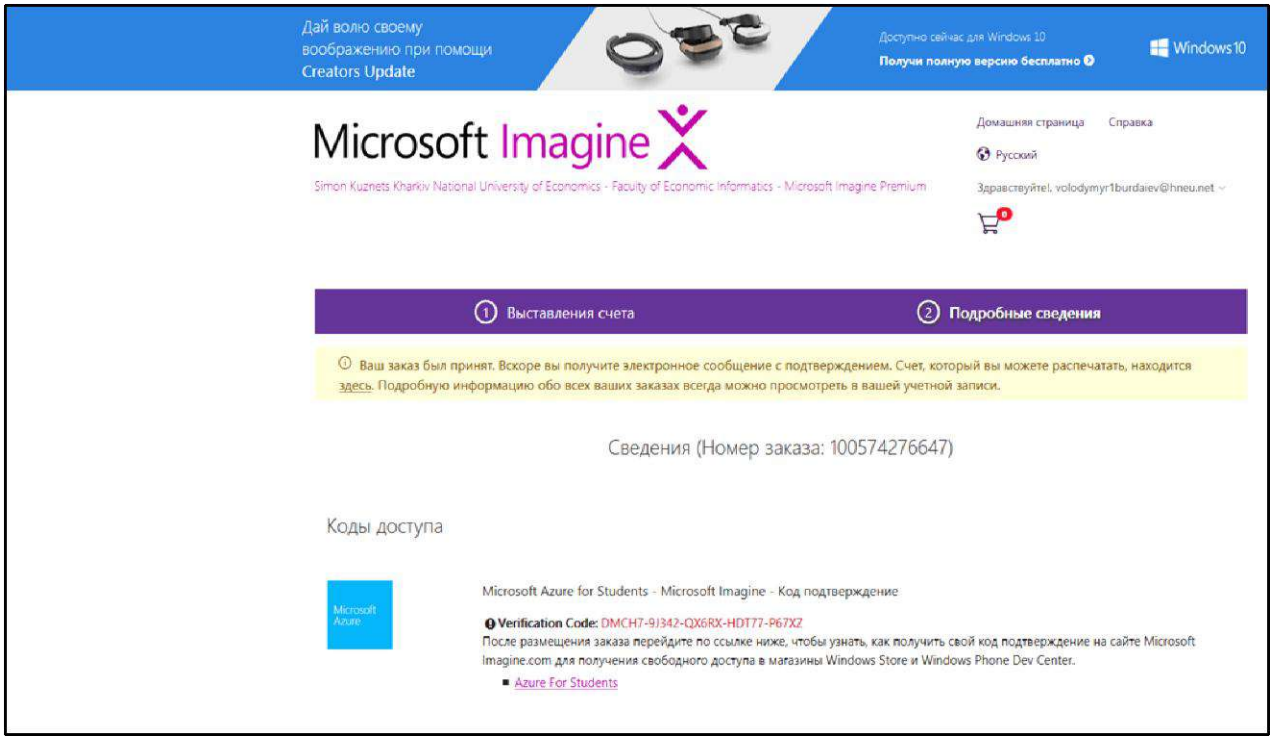


Рис. 8.2. **Вид вікна Microsoft Imagine**

3. Клацніть на посиланні "Sign in to Access" (рис. 8.3).

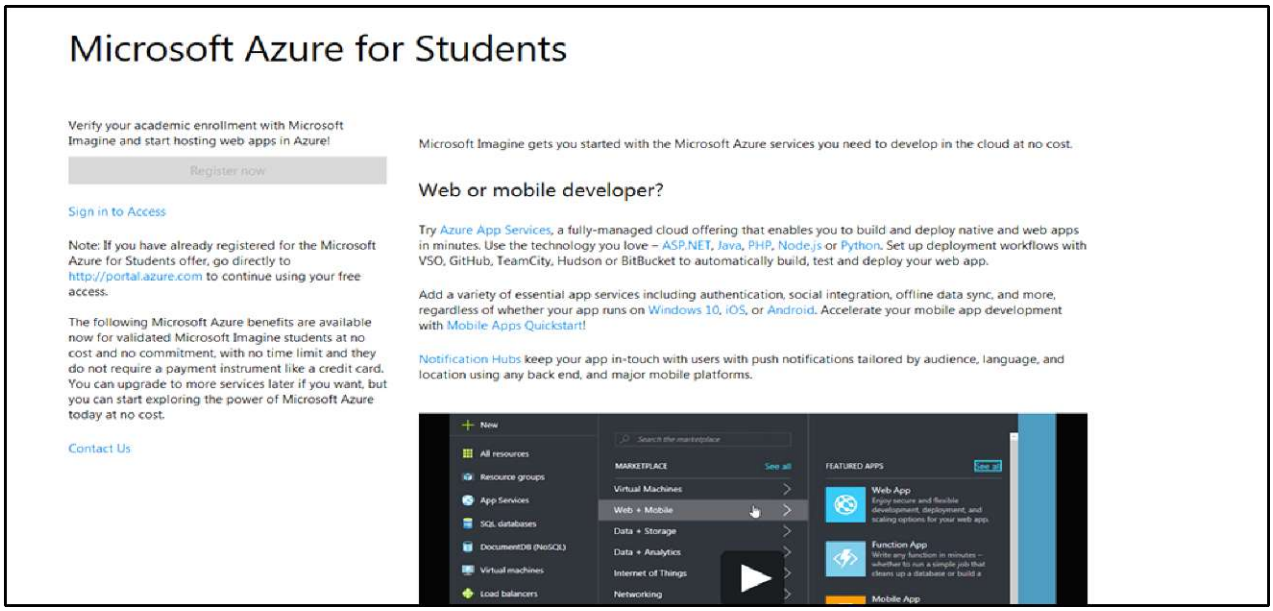


Рис. 8.3. **Вид вікна Microsoft Azure for Students**

4. Введіть пароль від Microsoft live id, заповніть форму та натисніть кнопку "Я згоден". Клацніть на посиланні "Please verify your student status", вставте код з Imagine (рис. 8.4).

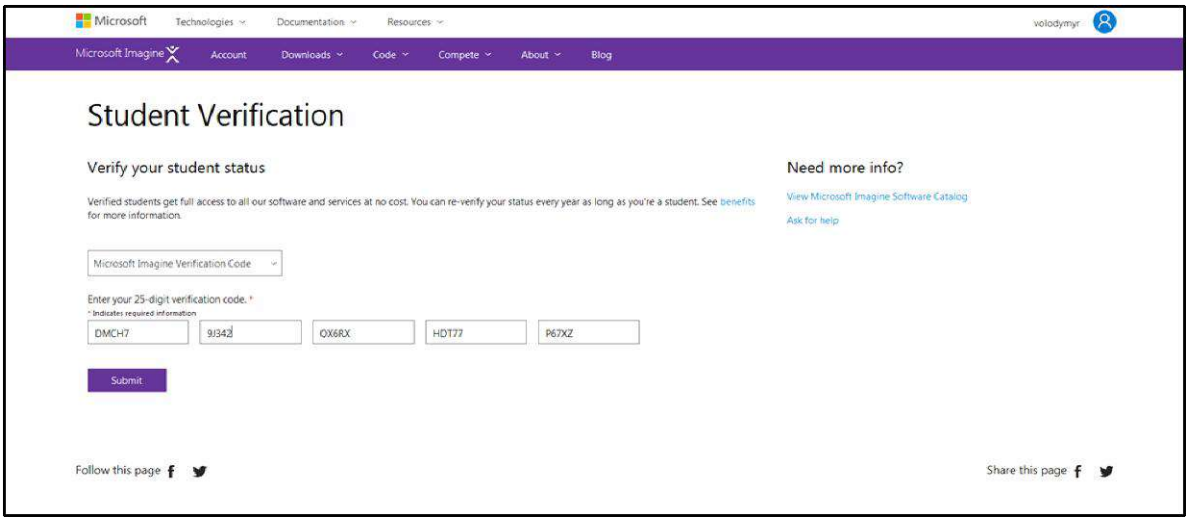


Рис. 8.4. **Вид вікна перевірки статусу студента**

5. Клацніть кнопку "Next" і посилання "Activate Azure"; клацніть посилання "Register Now". Введіть e-mail. Виберіть особистий обліковий запис. Виконайте реєстрацію Azure: заповніть форму, введіть свій номер мобільного телефону. Введіть код (отриманий у вигляді СМС). Підтвердіть код. Відзначте чек-бокс "Я приймаю угоду" та натисніть кнопку "Реєстрація". Зачекайте появи запрошення в Microsoft Azure (рис. 8.5).

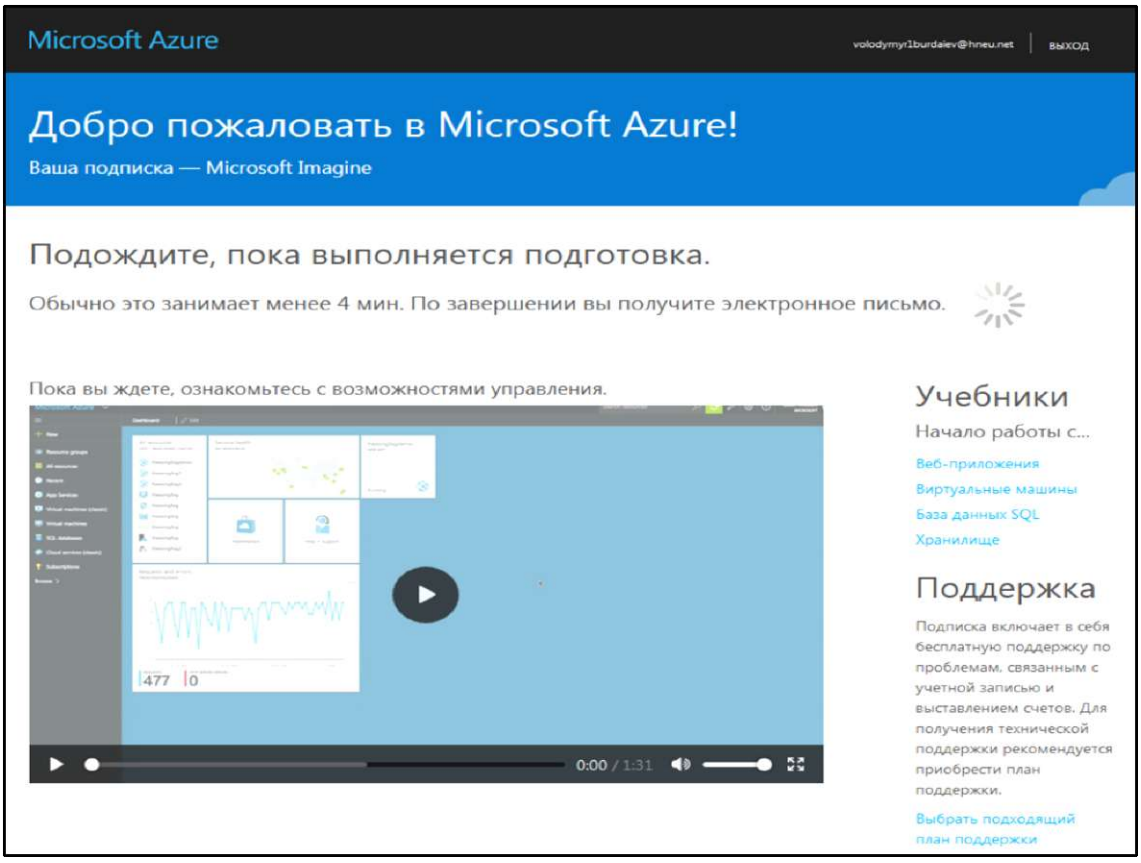


Рис. 8.5. **Вид вікна про запрошення в Microsoft Azure**

Примітка: підписку можна отримати безкоштовно на 30 днів, використовуючи, наприклад, кредитну або Інтернеткартку банку. Процес реєстрації аккаунта на порталі Microsoft Azure аналогічний. Тоді сервіс Microsoft Azure спочатку зніме з картки 1 долар, а після перевірки реквізитів, зазначених при реєстрації, назад поверне 1 долар.

8.2. Створення нової бази даних і виконання її розгортання на порталі Microsoft Azure

Завдання 2

Створіть порожню базу даних і виконайте її розгортання.

Виконання

Для виконання завдання необхідно дотримуватись такої послідовності дій.

1. Зайдіть на портал Azure за посиланням: https://portal.azure.com.
2. Виберіть активну підписку Azure for Students, якщо їх декілька.
3. Виберіть пункт меню **Бази даних SQL** на лівій панелі (рис. 8.6).

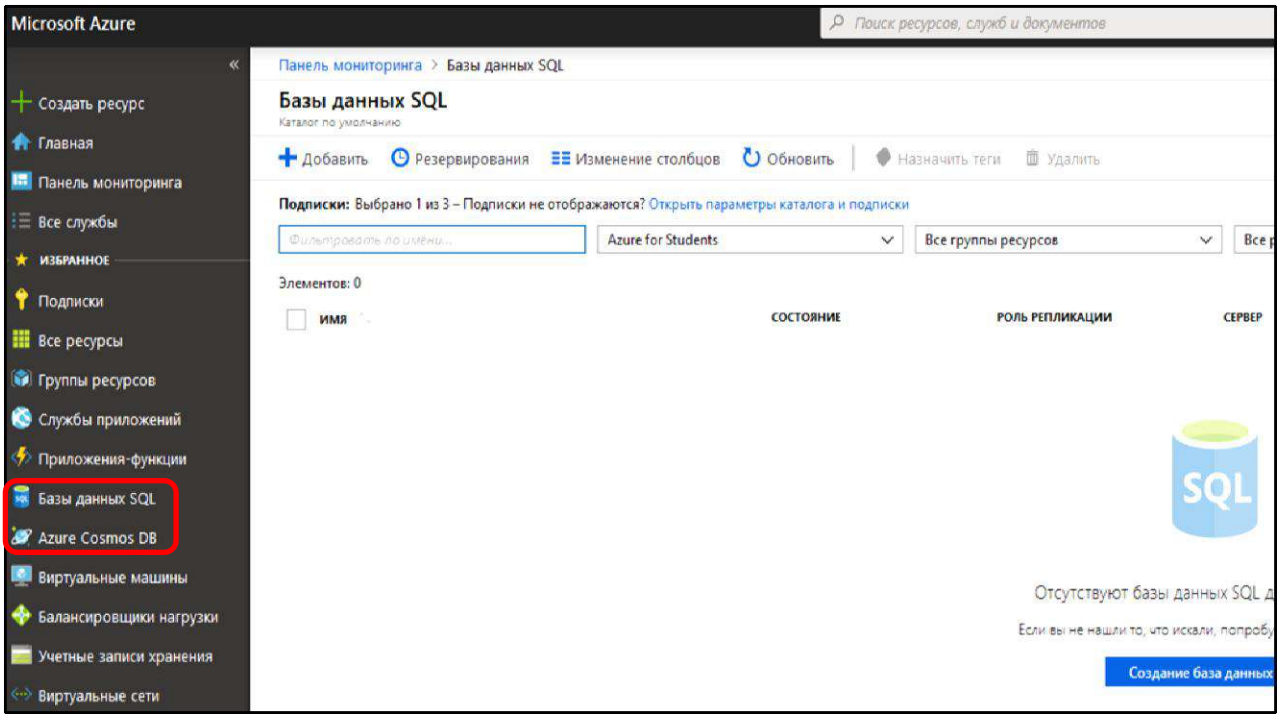


Рис. 8.6. **Створення нової бази даних**

4. Заповніть форму бази даних SQL, вказавши інформацію, показану на рис. 8.7.

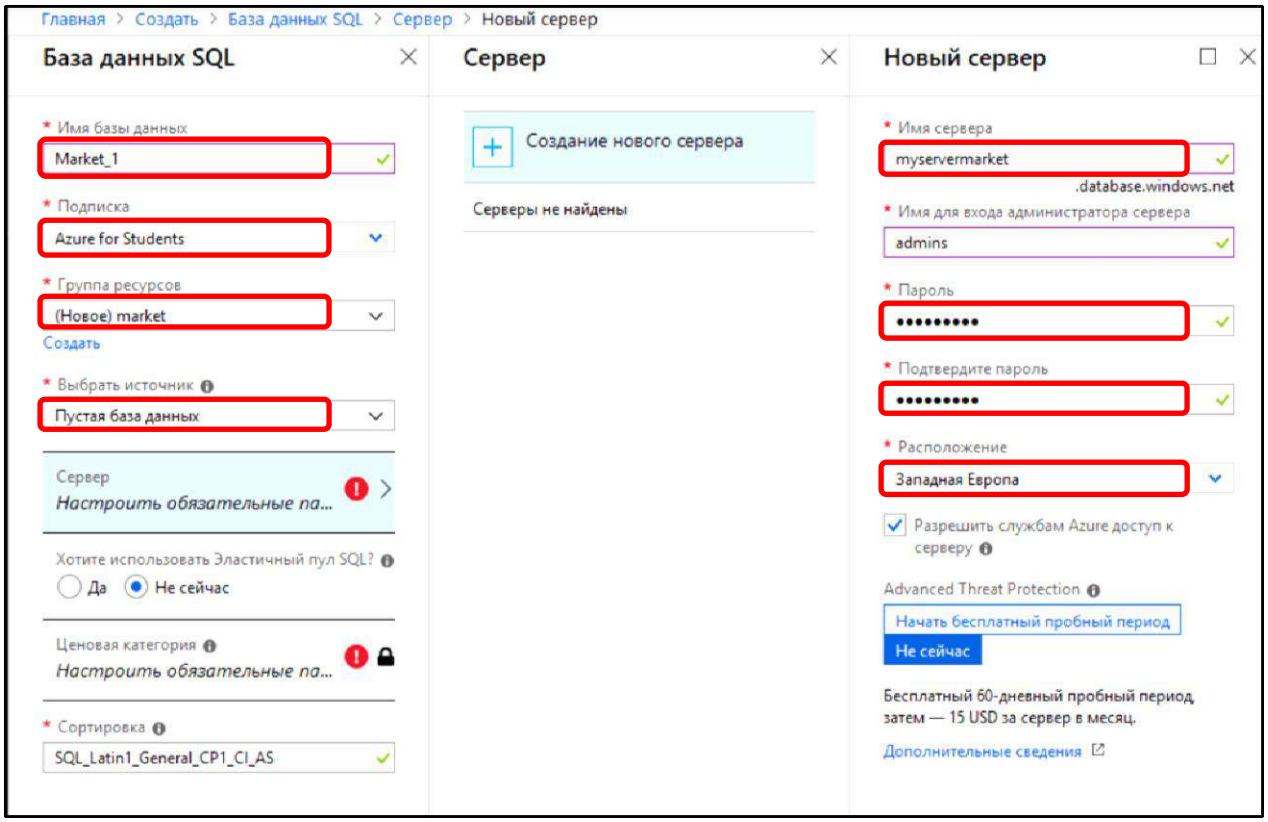


Рис. 8.7. **Інформація про нову базу даних і сервер**

5. Натисніть кнопки **Вибрати** та **Створити**. Далі Azure розгортає базу даних (рис. 8.8).

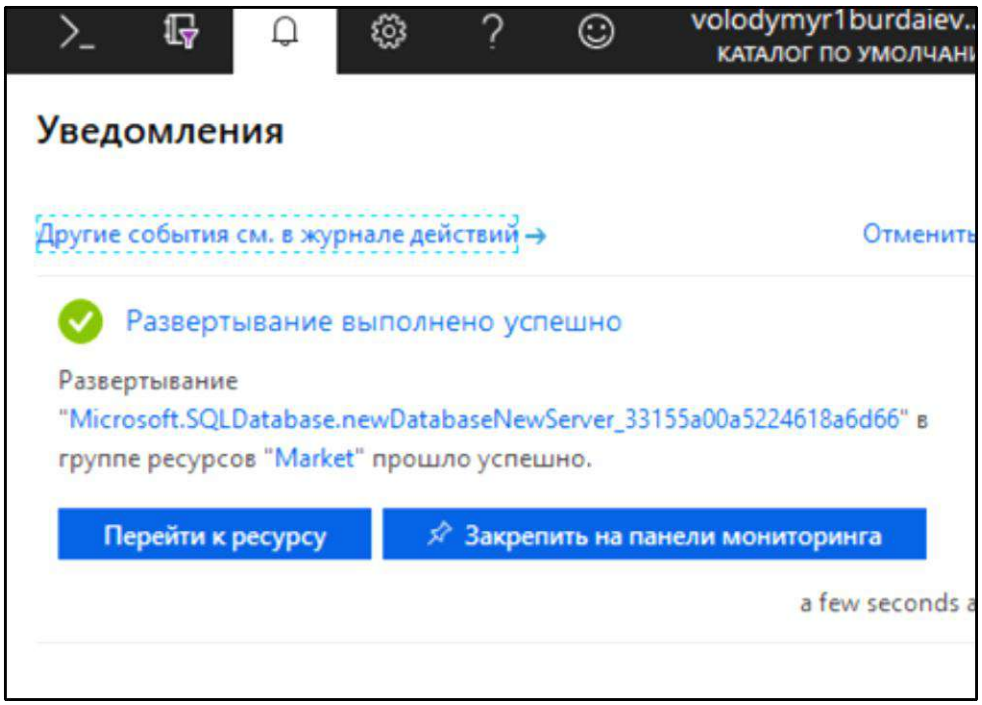


Рис. 8.8. **Повідомлення про розгортання нової бази даних**

Примітка: для заповнення форми бази даних SQL слід указати таку інформацію: database name – ім'я бази даних; підписка – ваша підписка; група ресурсів – ім'я групи ресурсів, де розташовується база даних. У розділі **Сервер** клацніть **Set** Необхідні параметри та заповніть форму SQL Server (логічний сервер), указавши таку інформацію: server name – ім'я сервера, ім'я для входу адміністратора сервера – будь-яке припустиме ім'я, пароль – будь-який допустимий пароль, місце розташування – будь-яке припустиме розташування.

8.3. Створення таблиці за допомогою редактора БД порталу Microsoft Azure

Завдання 3

Створіть таблицю **Client** у базі даних **Market_1**.

Виконання

Для виконання завдання необхідно дотримуватись такої послідовності дій.

1. У лівій області меню клацніть **Бази даних SQL**, потім виберіть **Market_1** і **Редактор запитів** і клацніть **Ввійти** (рис. 8.9).

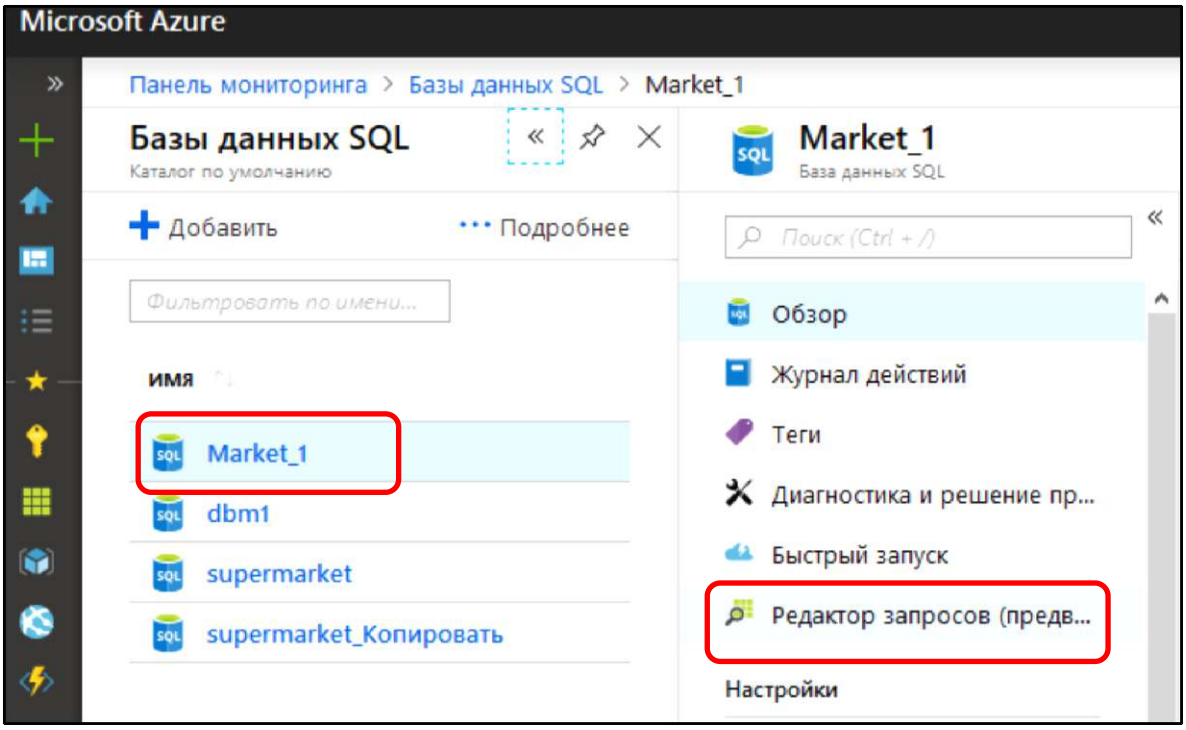


Рис. 8.9. **Вибір редактора запитів**

2. Виберіть аутентифікацію SQL Server, укажіть необхідні облікові дані та натисніть кнопку **ОК**, щоб виконати вхід.

3. Після того як буде пройдена перевірка, що ви є дійсно адміністратор сервера, в області редактора запитів введіть запит для створення таблиці **Client** і натисніть кнопку **Запустити** (рис. 8.10).

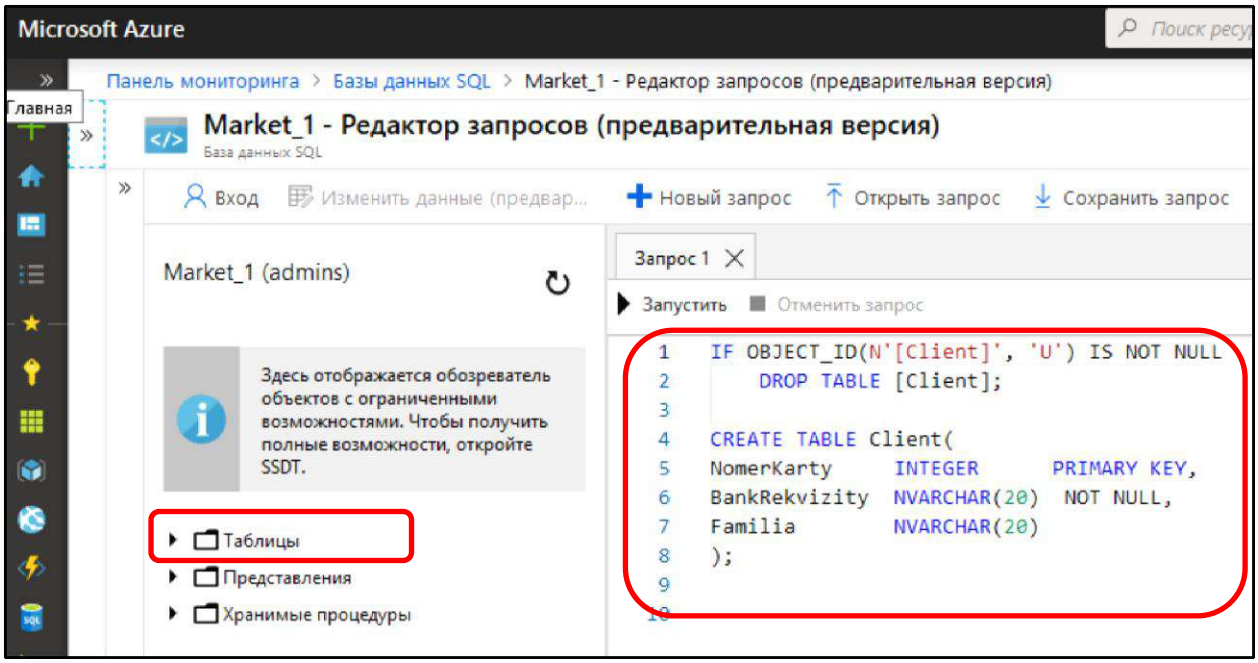


Рис. 8.10. **Запит для створення таблиці Client**

8.4. Виконання запитів до БД

Завдання 4

Створіть запити до бази даних **Market_1**.

Виконання

Для виконання завдання необхідно дотримуватись такої послідовності дій.

1. Пройдіть перевірку, що ви є дійсно адміністратор сервера. В області редактора запитів введіть запит для створення таблиць: **Departament, Goods, Desk, Realization** і натисніть кнопку **Запустити** (рис. 8.11).

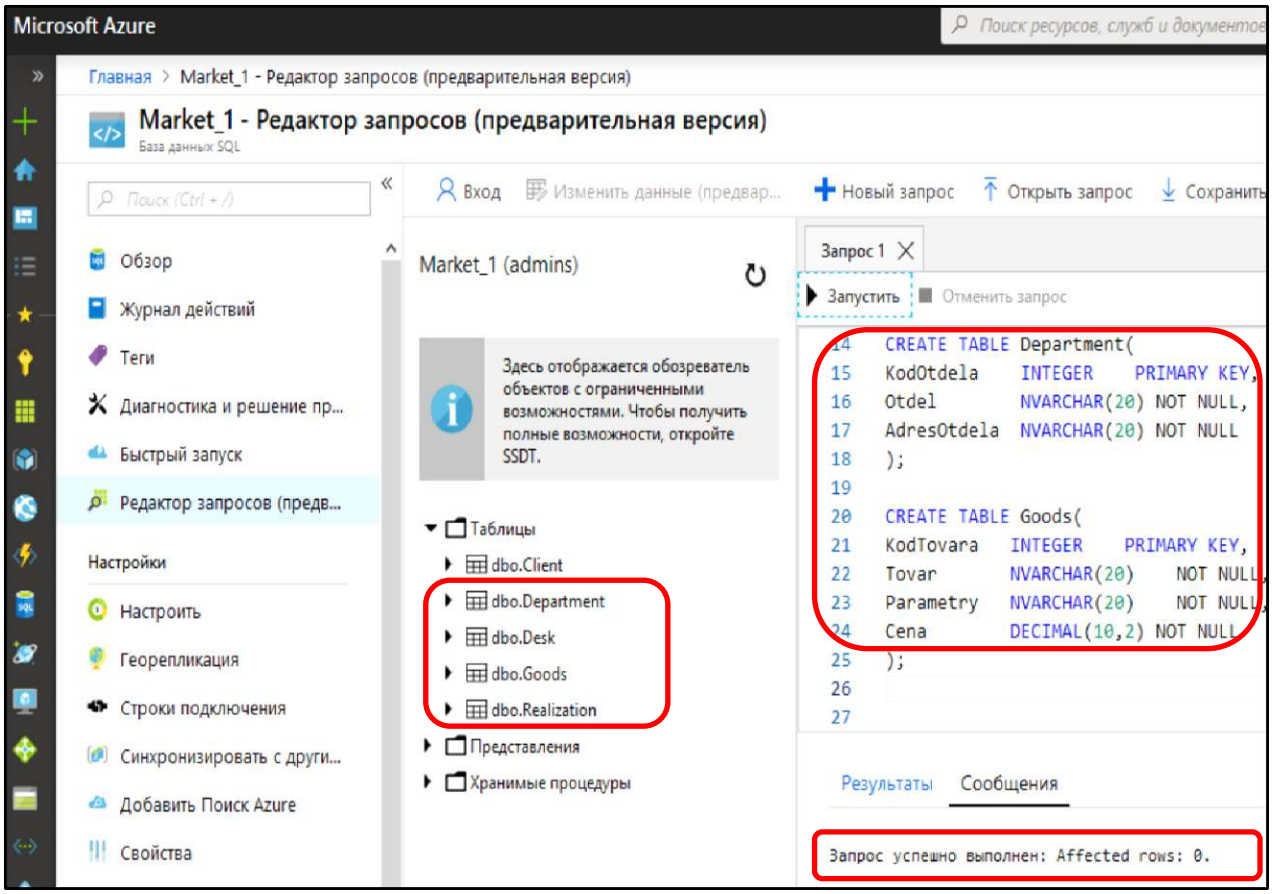


Рис. 8.11. **Результат запиту на створення таблиць Departament, Goods, Desk, Realization**

2. Використайте інструкцію Transact-SQL INSERT, щоб додати нові записи в таблиці: **Client, Departament, Goods, Desk, Realization** і натисніть кнопку **Запустити** (рис. 8.12).

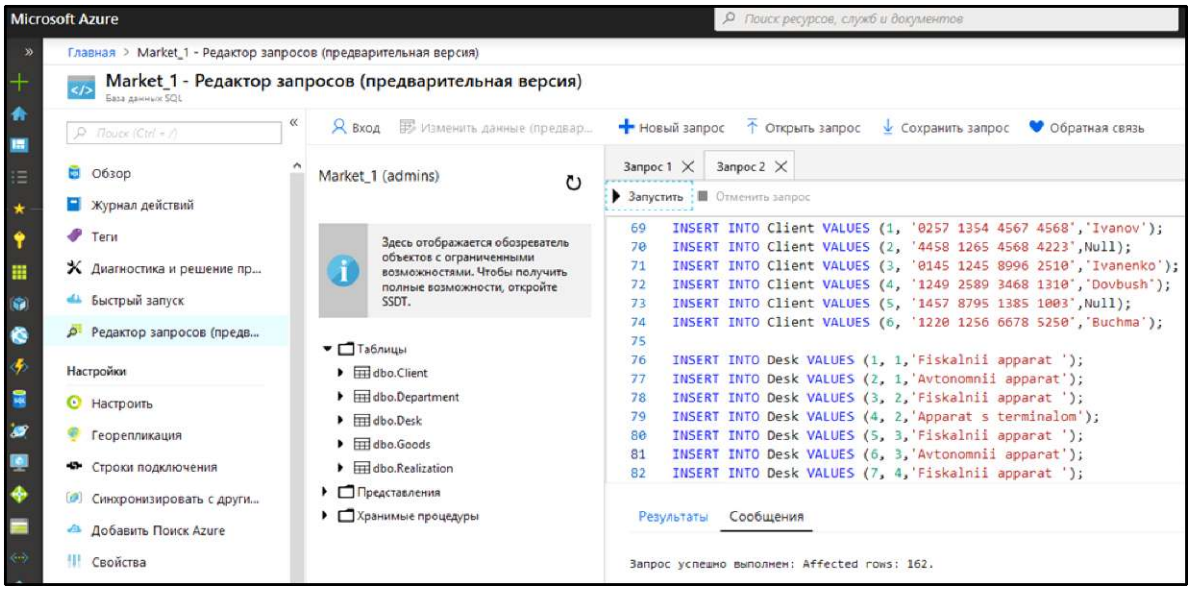


Рис. 8.12. **Вид запиту на заповнення записами таблиць Client, Departament, Goods, Desk, Realization**

3. Введіть запит на перегляд вмісту таблиці **Client** і натисніть кнопку **Запустити** (рис. 8.13).

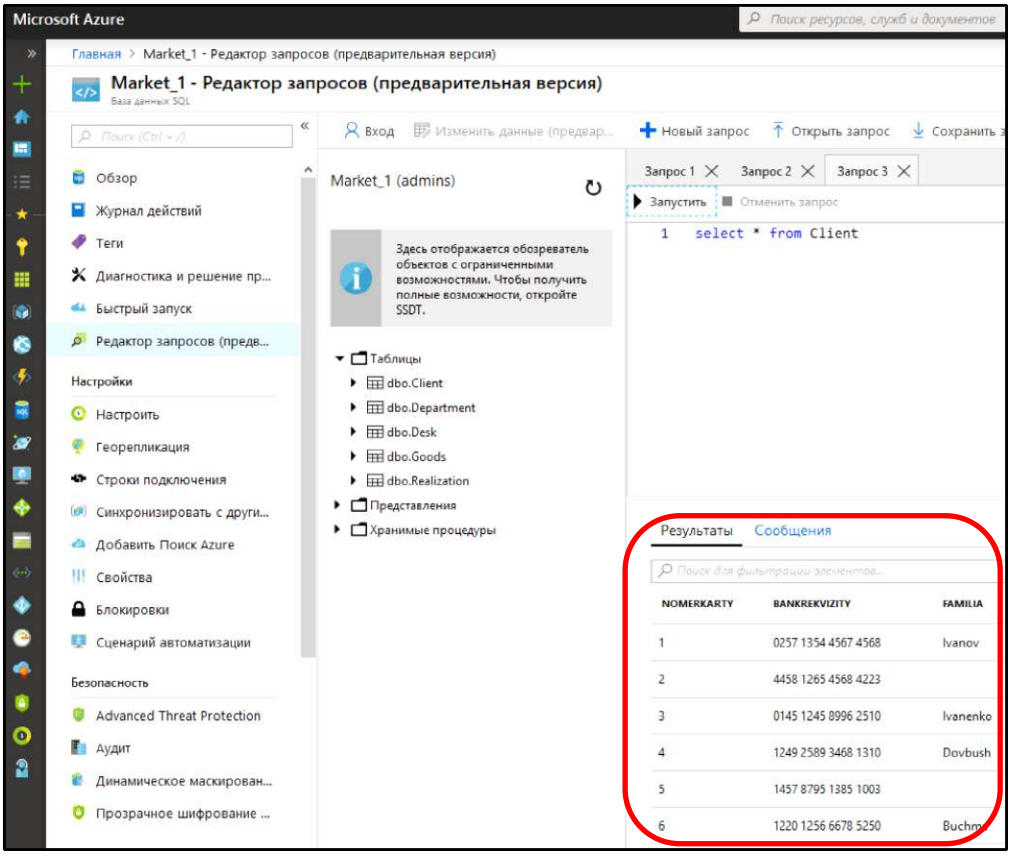


Рис. 8.13. **Усі записи таблиці Client**

Примітка: таблиці взяті з контрольної роботи першого модуля (база даних: Супермаркет.mdf).

Можна також виконувати різні запити на оновлення і видалення записів. Аналогічним чином можна створити подання і процедури (лабораторні роботи 1 і 2).

8.5. Виконання міграції локальної БД на портал Microsoft Azure

Завдання 5

Виконайте міграцію локальної бази даних **Supermarket_1** на портал Microsoft Azure.

Виконання

Для виконання завдання необхідно дотримуватись такої послідовності дій.

1. Створіть порожню базу даних **Supermarket_1** на порталі Microsoft Azure (рис. 8.14).

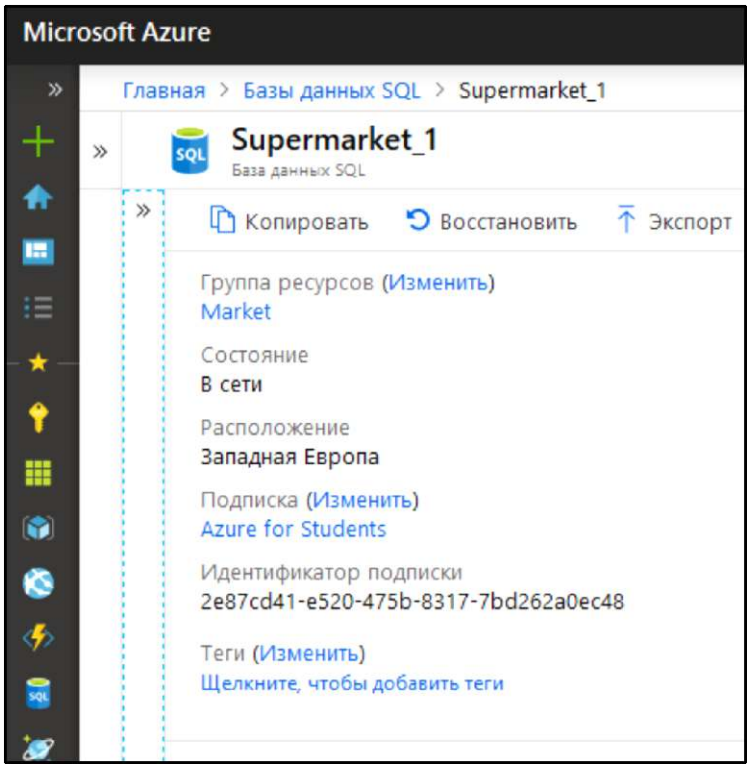


Рис. 8.14. **Інформація про базу даних Supermarket_1**

2. Створіть правило брандмауера на рівні сервера БД SQL, що дозволяє зовнішнє підключення через брандмауер БД тільки з певної IP-адреси. У вікні створеної БД виберіть пункт **Налаштування брандмауера.** Введіть: ім'я правила, початкову і кінцеву IP-адресу (рис. 8.15).

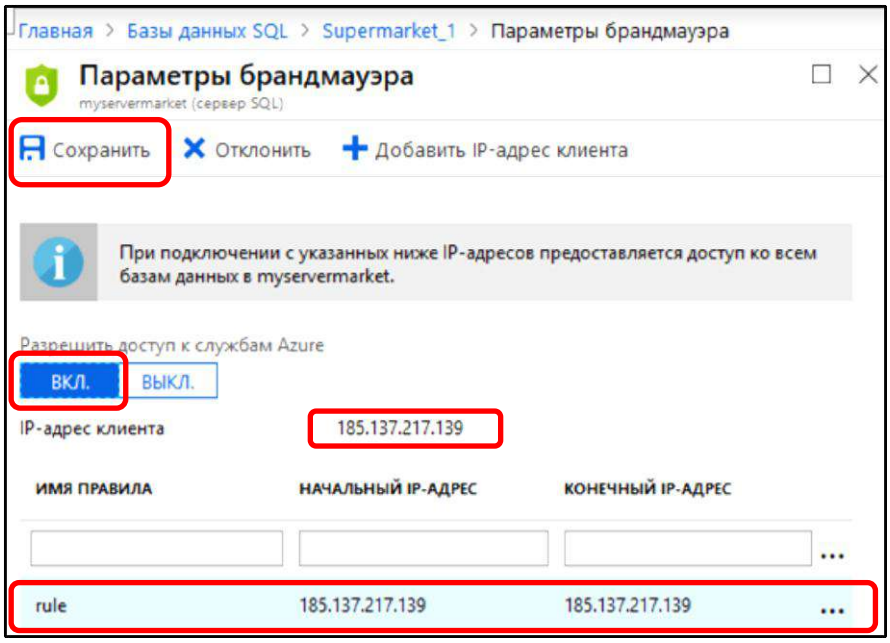


Рис. 8.15. **Доступ до бази даних з IP-адреси**

3. Виконайте міграцію бази даних **Supermarket** з Visual Studio 2017 (рис. 8.16) у базу даних **Supermarket_1** на порталі Microsoft Azure за допомогою програми Microsoft Data Migration Assistant (сайт програми https://www.microsoft.com/en-us/download/confirmation.aspx?id=53595).

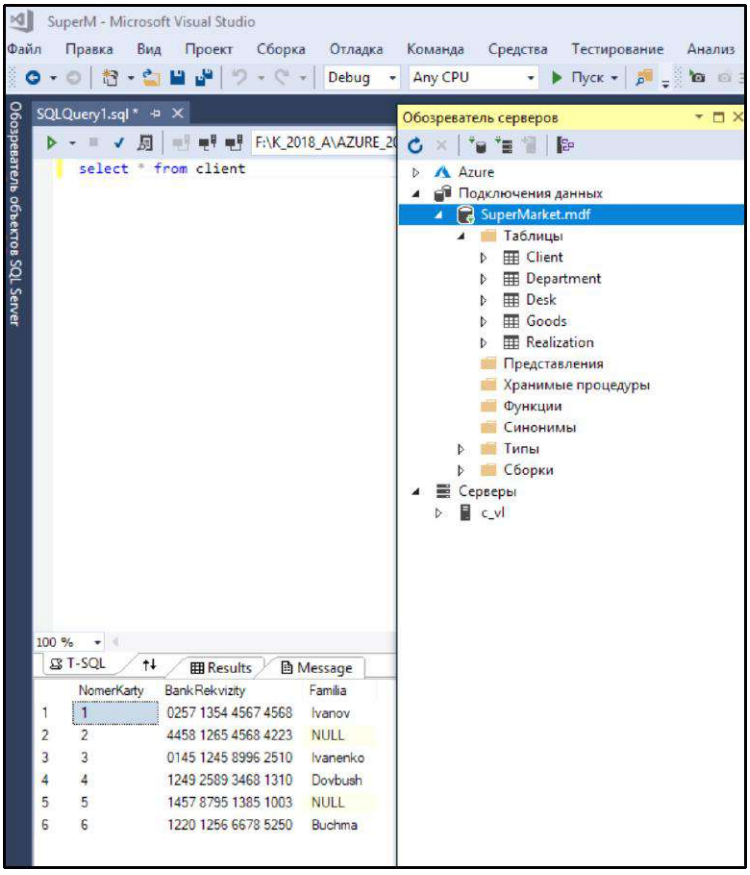


Рис. 8.16. **База даних Supermarket в Visual Studio 2017**

4. Установіть програму Microsoft Data Migration Assistant на комп'ютер, де знаходиться база даних Supermarket. Для інсталяції програми погодьтеся з усіма пропозиціями. Відкрийте Data Migration Assistant. Для створення проекту в меню зліва клацніть "+ Створити".

5. Виберіть тип проекту – Supermarket_Migration. Введіть назву проекту. Виберіть: тип вихідного сервера – SQL Server, тип цільового сервера – Azure SQL Database, область перенесення – Schema and data. Натисніть кнопку **Create** (рис. 8.17).

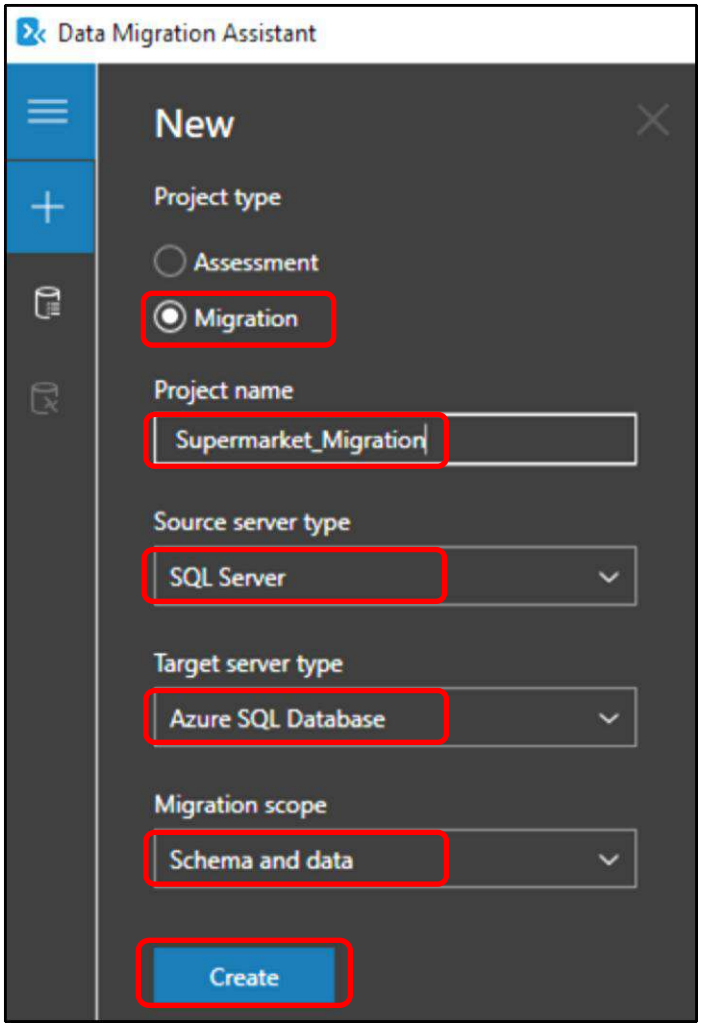


Рис. 8.17. **Налаштування проекту**

6. Введіть ім'я вихідного сервера в поле Server name – (localdb) \ MSSQLLocalDB. Виберіть тип перевірки автентичності – Windows Authentication. Натисніть кнопку **Connect**. Виберіть необхідну вихідну БД – **SUPERMARKET.MDF**. Натисніть кнопку **Next** (рис. 8.18).

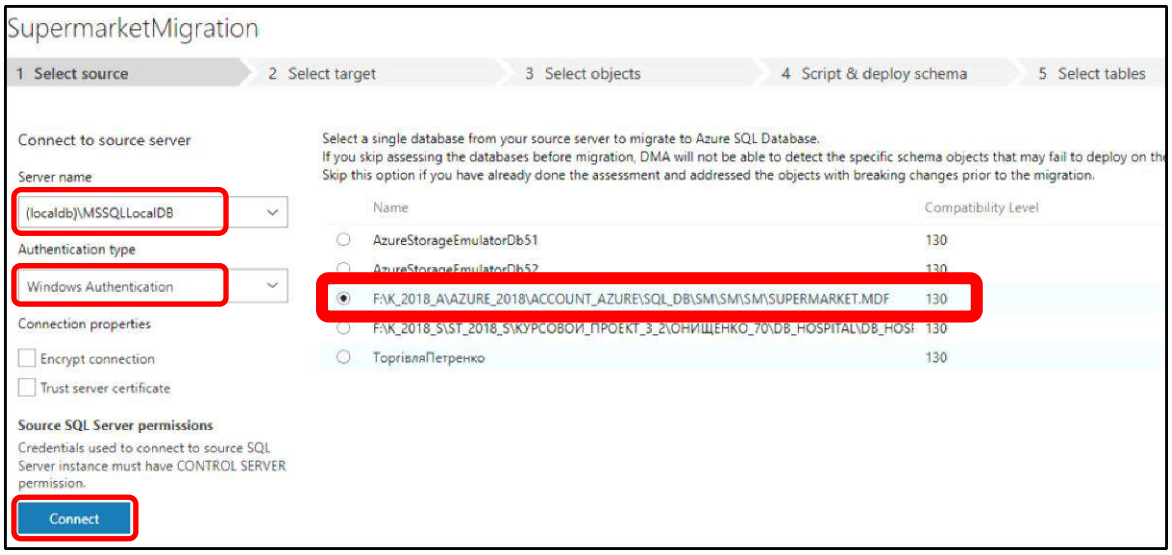


Рис. 8.18. **Вибір БД – SUPERMARKET.MDF**

7. Введіть повне ім'я сервера – myservermarket. database. windows.net БД Azure. Виберіть тип перевірки автентичності – SQL Server, ім'я користувача – admins і пароль для підключення з сервера. Позначте чек-бокси для пунктів **Encrypt** connection і **Trust server certificate**. Виберіть необхідну цільову БД Azure – **Supermarket_1**. Натисніть кнопку **Next** (рис. 8.19).

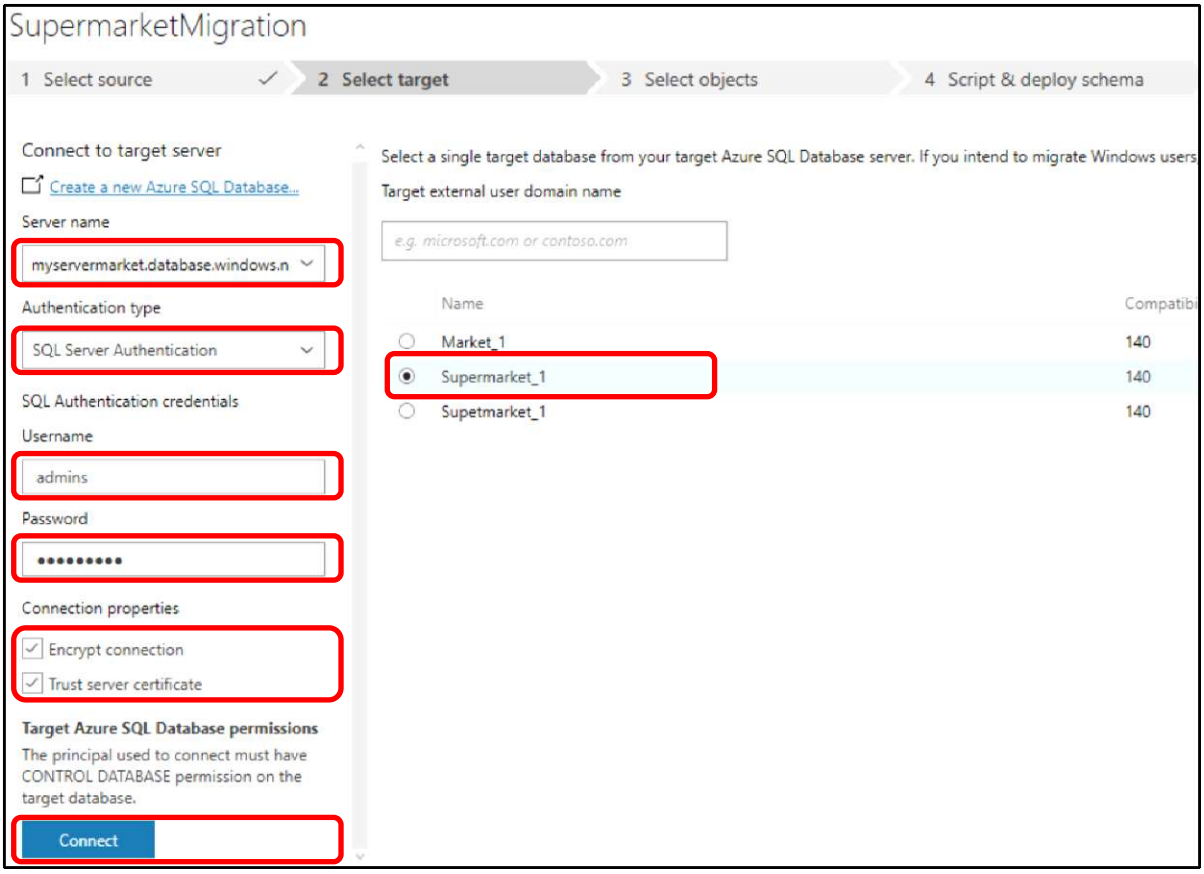


Рис. 8.19. **Вибір БД – Supermarket_1 на Azure**

8. Виберіть об'єкти з вихідної БД для перенесення в цільову БД. Клацніть кнопку **Generate SQL Script** і кнопку **Deploy Schema**. Виберіть таблиці для міграції і клацніть кнопку **Start data migration**. Результати міграції БД показані на рис. 8.20.

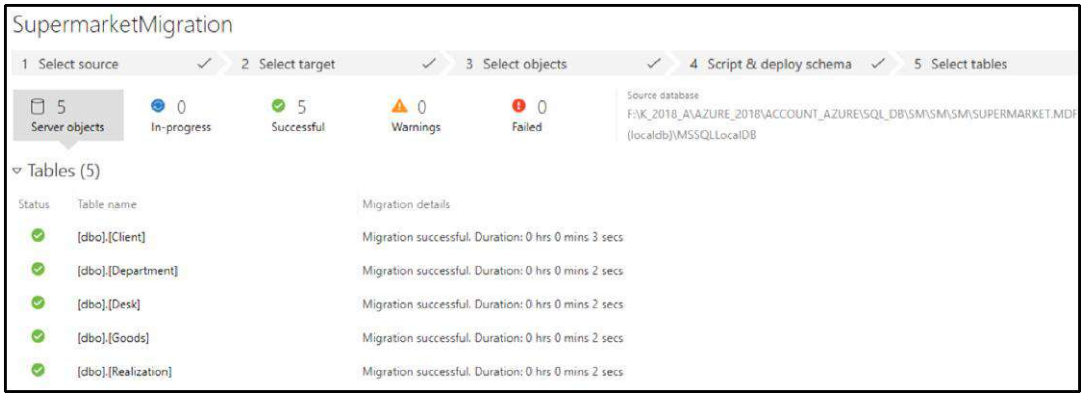


Рис. 8.20. **Migrate data**

9. У Azure база даних **Supermarket_1** має вигляд, зображений на рис. 8.21.

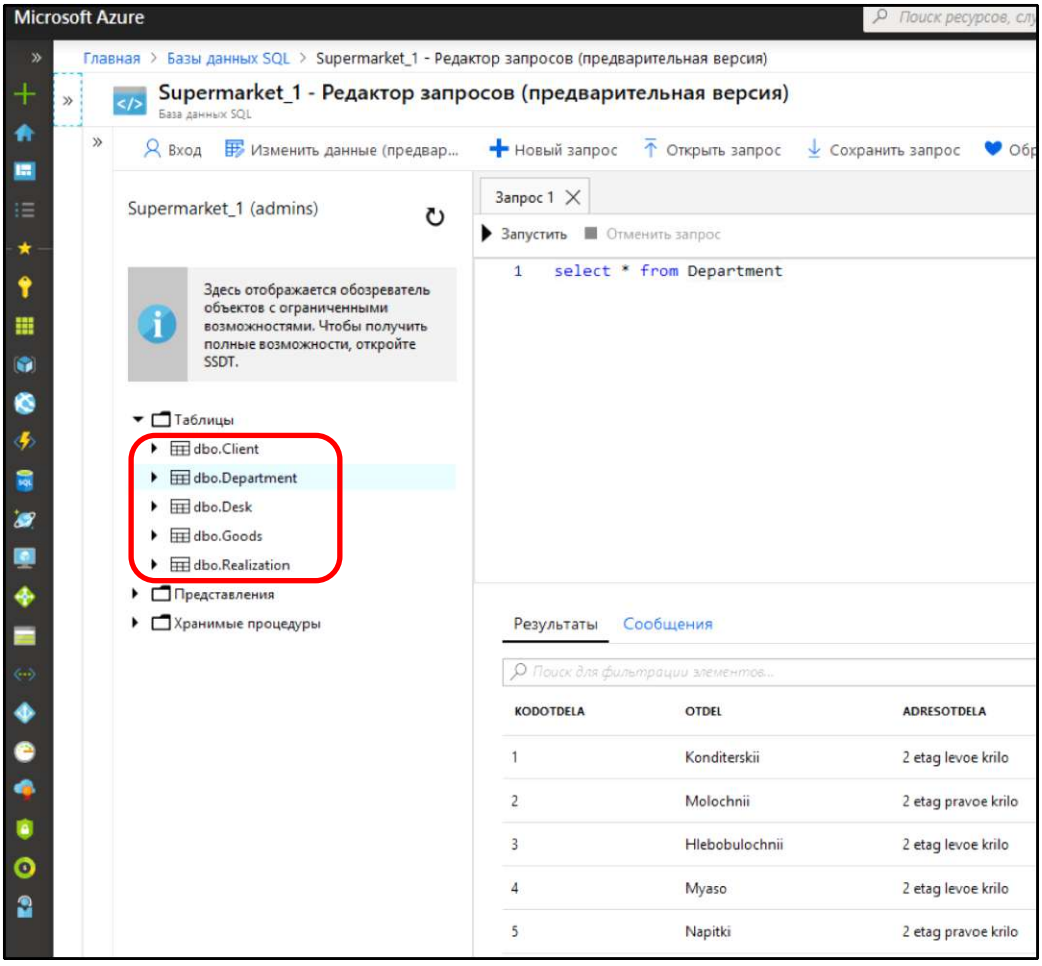


Рис. 8.21. **Вид бази даних Supermarket_1 на Azure**

Завдання для самостійного виконання

1. Виконайте міграцію бази даних Супермаркет (файл Супермаркет.mdf) на портал Azure.

2. Визначте банківські реквізити клієнтів і загальну кількість їх купівель.

3. Визначте середню ціну тортів.

4. Виведіть назви товарів і загальну суму реалізації кожного товару з м'ясного відділу.

5. Визначте назви товарів, номери кас і типи касових апаратів, де оформлялися купівля сирів і печива.

6. Для кожного відділу визначте його назву, місце розташування і загальну суму реалізованих товарів.

7. Визначте перелік і ціну товарів, реалізованих у відділах 2, 4 і 5.

8. Визначте усі товари, у яких в назві третя буква "t" і ціна яких більше 5 грн.

9. Визначте відділ, в якому був проданий товар з найбільшою кількістю закупівель.

10. Виведіть усі відомості про цукерки за зростанням їхньої ціни.

11. Визначте назви відділів, у яких були продані товари, що починаються на літеру "С".

12. Виведіть назви та загальну суму реалізації кожного товару з кондитерського відділу.

13. Для кожного відділу визначте загальну суму всіх купівель.

14. Визначте назви відділів і середню ціну проданих товарів у кожному відділі.

15. Для кожного клієнта визначте товар, куплений ним на максимальну суму. Упорядкуйте результати за вартістю.

16. Визначте назви товарів, номери кас і типи касових апаратів, де оформлялися купівлі.

17. Для кожного відділу визначте його назву, місце розташування і кількість реалізованих товарів

18. Виконайте міграцію бази даних за індивідуальним завданням із лабораторної роботи 4.

Рекомендована література

1. Андон Ф. Язык запросов SQL : учебный курс / Ф. Андон, В. Резниченко. – Санкт-Перербург : Питер ; Киев : BHV, 2006. – 416 с.

2. Астахова И. Ф. SQL в примерах и задачах / И. Ф. Астахова, А. П. Толстобров, И. М. Мельников. – Минск : Новое знание, 2002. – 176 с.

3. Барсегян А. А. Методы и модели анализа данных: OLAP и Data Mining / А. А. Барсегян, М. С. Куприянов, В. В. Степаненко. – Санкт-Петербург: БХВ-Петербург, 2004. – 336 с.

4. Бен-Ган Ицик. Microsoft SQL Server 2012. Основы T-SQL / Ицик Бен-Ган. – Москва : Эксмо, 2015. – 400 с.

5. Бондарь А. Microsoft SQL Server 2014 / А. Бондарь. – Санкт-Петербург : БХВ-Петербург, 2015. – 592 с.

6. Боуман Д. Практическое руководство по SQL / Д. Боуман, С. Эмерсон, М. Дарновски. – Москва : Вильямс, 2002. – 352 с.

7. Бэнкер К. MongoDB в действии = MongoDB in Action. /К. Бєнкер. – ДМК Пресс, 2014. – 394 с.

8. Дейт Дж. Введение в системы баз данных / Дж. Дейт. – 8-е изд. – Москва : Вильямс, 2005. – 1328 с.

9. Руководство по MongoDB [Электронный ресурс]. – Режим доступа : https://metanit. com/nosql /mongodb/.

10. Степанов В. П. Лабораторний практикум з навч. дисц. "Бази даних кінцевих користувачів" : навч.-практ. посіб. / В. П. Степанов, В. П. Бурдаєв, Т. В. Донченко та ін. – Харків : Вид. ХНЕУ, 2012. – 228 с.

11. Тарасов О. В. Організація баз даних та знань. Проектування баз даних : навч.-практ. посіб. для самостійної підготовки студентів. Ч. 1 / О. В. Тарасов, В. В. Федько, М. Ю. Лосєв. – Харків : Вид. ХНЕУ, 2011. – 198 с.

12. Тарасов О. В. Використання мови SQL для роботи з сучасними системами управління базами даних / О. В. Тарасов, М. Ю. Лосєв, В. В. Федько. – Харків : Вид. ХНЕУ, 2013. – 348 с.

13. Тарасов О. В. Клієнт-серверні технології СУБД Oracle. Мова SQL Oracle. навч. посіб. для самостійної підготовки студентів з навч. дисц. "Організація баз даних та знань" / О. В.Тарасов, В. В. Федько. – Харків : Вид. ХНЕУ ім. С. Кузнеця, 2015. – 384 с.

14. Федько В. В. Лабораторний практикум "Основи баз даних та знань" з модуля навч. дисц. "Організація баз даних та знань": Навч.-практ. посіб. / укл. В. В. Федько, О. В. Тарасов, М. Ю. Лосєв. – Харків : Вид. ХНЕУ, 2011. – 192 с.

15. Фленов М. Transact-SQL [Электронный ресурс] / М. Фленов. – Режим доступа : www.litres.ru/mihail-flenov/transact-sql-2900725/.

16. Форта Б. Освой самостоятельно SQL / Б. Форта. – Москва : Вильямс, 2005. – 288 с.

17. Hows D. The Definitive Guide to MongoDB: A complete guide to dealing with Big Data using MongoDB / D. Hows, E. Plugge, P. Membrey, T. Hawkins. – Apress, 2013. – 336 p.

18. Kimball R. The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling, Third Edition / R. Kimball, M. Ross. – John Wiley & Sons, Inc, 2013. – 564 p.

19. Transact-SQL Reference [Electronic resource]. – Access mode : https://docs.microsoft.com/en-us/sql/t-sql/language-reference?view=sql-server-2017

20. IDEF1X Data Modeling Method [Електронний ресурс] – Режим доступу : http://www.idef.com/IDEF1x.htm.

21. Business Intelligence Blogs. [Electronic resource]. – Access mode : https://blogs.msdn.microsoft.com/business-intelligence/.

22. Microsoft Azure Options for SQL Server Relational Databases. [Electronic resource]. – Access mode : https://azure microsoft.com/ru-ru/ resources/microsoft-azure-options-for-sql-server-relational-databases.

23. MongoDB Atlas. [Electronic resource]. – Access mode: https://www. mongodb.com/.

Зміст

НАВЧАЛЬНЕ ВИДАННЯ

Федько Віктор Васильович
Бурдаєв Володимир Петрович

БАЗИ ДАНИХ

Лабораторний практикум
для студентів галузі знань 12 "Інформаційні технології" першого (бакалаврського) рівня

Самостійне електронне текстове мережеве видання

Відповідальний за видання *О. Г. Руденко*

Відповідальний редактор *М. М. Оленич*

Редактор *Н. І. Ганцевич*

Коректор *Т. А. Маркова*

План 2019 р. Поз. № 82 ЕВ. Обсяг 215 с.

Видавець і виготовлювач – ХНЕУ ім. С. Кузнеця, 61166, м. Харків, просп. Науки, 9-А

Свідоцтво про внесення суб'єкта видавничої справи до Державного реєстру
ДК № 4853 від 20.02.2015 р.