

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»**

СЕРІЯ «КІБЕРБЕЗПЕКА ТА ШТУЧНИЙ ІНТЕЛЕКТ»

**А. А. Гаврилова, С. С. Погасій, Р. В. Корольов,
В. С. Хвостенко, Т. С. Мілевська**

ТЕХНОЛОГІЇ БАЗ ДАНИХ

Навчально-практичний посібник

За загальною редакцією доктора технічних наук, професора С. П. Євсєєва

Львів
«Новий Світ-2000»
2025

УДК 004.056(075.8)

I 76

*Рекомендовано вченою радою
Національного технічного університету «Харківський політехнічний інститут»
Протокол № 2 від 21.02.2025 р.*

Укладачі:

*А. А. Гаврилова, доцентка кафедри кібербезпеки НТУ «ХПІ»;
С. С. Погасій, доцент кафедри кібербезпеки НТУ «ХПІ»;
Р. В. Корольов, доцент кафедри кібербезпеки НТУ «ХПІ»;
В. С. Хвостенко, доцент кафедри кібербезпеки НТУ «ХПІ»;
С. В. Мілевський, доцент кафедри кібербезпеки НТУ «ХПІ».*

Рецензенти:

*І. Р. Опірський, доктор технічних наук, професор, завідувач кафедри захисту інформації Національного університету «Львівська політехніка»;
С. Е. Остапов, доктор фізико-математичних наук, професор кафедри програмного забезпечення комп'ютерних систем Чернівецького національного університету імені Юрія Федьковича.*

Гаврилова А. А., Погасій С. С., Корольов Р. В., Хвостенко В. С., Мілевська Т. С.

I 76 Технології баз даних : навчально-практичний посібник / уклад. А. А. Гаврилова, С. С. Погасій, Р. В. Корольов, В. С. Хвостенко, Т. С. Мілевська ; за заг. ред. С. П. Євсєєва. – Харків : НТУ «ХПІ», – Львів : «Новий Світ-2000», 2025. – 222 с. – (Серія «Кібербезпека та штучний інтелект»).

ISBN 978-966-418-516-2

Навчально-практичний посібник присвячений основним питанням навчальної дисципліни «Організація та безпека баз даних», яка викладається на 4 курсі бакалаврата за освітньою програмою «Кібербезпека». Запропоновано матеріали лекційних занять та методичні вказівки з лабораторного практикуму за навчальною програмою дисципліни.

Практична частина реалізована через виконання практичних завдань в лабораторній частині посібника та закріплюється шляхом самостійного виконання завдань за варіантами в частині «Завдання для виконання підсумкової роботи».

Посібник може бути використано здобувачами спеціальності 125 «Кібербезпека та захист інформації» денної та заочної форми навчання.

УДК 004.056(075.8)

© Гаврилова А. А., Погасій С. С., Корольов Р. В.,
Хвостенко В. С., Мілевська Т. С., 2025
© За заг. ред. Євсєєва С. П., 2025
© НТУ «ХПІ», 2025
© Видавництво ПП «Новий Світ – 2000»,
ФОП Піча С. В., 2025

ISBN 978-966-418-516-2

ЗМІСТ

ПЕРЕДМОВА	6
ЛЕКЦІЙНИЙ МАТЕРІАЛ.....	8
1. РЕЛЯЦІЙНА МОДЕЛЬ ДАНИХ.....	8
1.1 Структурна частина реляційної бази даних	8
1.2 Стандарти SQL	8
1.3 Типи даних в SQL. Перетворення типів	9
1.4 SQL-операції.....	11
2. ФОРМУВАННЯ SQL-ЗАПИТУ НА ВИБІРКУ ДАНИХ.....	12
2.1 Визначення SQL-запиту	12
2.2 Оператори SQL.....	17
2.2.1 Арифметичні оператори SQL	17
2.2.2 Побітові оператори SQL.....	17
2.2.3 Оператори порівняння SQL	17
2.2.4 Складені оператори SQL	17
2.2.5 Логічні оператори SQL.....	18
2.2.6 Оператори визначення даних.....	18
2.2.7 Оператори маніпуляції даними	18
2.2.8 Оператори визначення доступу до даних.....	18
2.2.9 Оператори управління транзакціями	18
3. БАЗА ДАНИХ SQL.....	19
3.1 Створення бази даних	19
3.2 Усунення надлишковості вибраних даних.	19
3.3 Уточнення запиту за допомогою предикатів	20
3.4 Спеціальні SQL-предикати. Належність до множини	20
3.5 Предикат діапазону.....	21
3.6 Предикат шаблону.....	21
3.7 Перевірка на значення NULL	21
4. ГРУПУВАННЯ ТА СОРТУВАННЯ ДАНИХ. ВИКОРИСТАННЯ ПІДЗАПИТІВ.....	23

4.1 Групування записів за значеннями одного або декількох стовпців	23
4.2 Використання агрегатних функцій.....	23
4.3 Використанням кванторів	25
4.4 Використання предиката EXISTS	26
5. ФОРМУВАННЯ СКЛАДНИХ SQL-ЗАПИТІВ.....	27
5.1 Використання теоретико-множинних операцій – формування складних SQL-запитів з використанням оператора INNER JOIN.....	27
5.2 Використання теоретико-множинних операцій – формування складних SQL-запитів з використанням оператора LEFT JOIN	27
5.3 Використання теоретико-множинних операцій – формування складних SQL-запитів з використанням оператора RIGHT JOIN.....	28
5.4 Використання теоретико-множинних операцій – формування складних SQL-запитів з використанням оператора FULL OUTER JOIN	29
5.5 Використання теоретико-множинних операцій – формування складних SQL-запитів з використанням оператора UNION та UNION ALL	30
6. ПРАВИЛА МАНІПУЛЯЦІЇ З ТАБЛИЦЯМИ ТА МОДИФІКАЦІЇ ДАНИХ.....	31
6.1 Створення таблиць.....	31
6.2 Обмеження на дані.....	31
6.3 Модифікація таблиць.....	33
6.4 Видалення таблиць.....	33
6.5 Внесення даних	34
6.6 Зміна даних	34
6.7 Видалення даних	35
7. ВІРТУАЛЬНІ ТАБЛИЦІ (VIEW – ПРЕДСТАВЛЕННЯ).....	36
7.1 Створення віртуальних таблиць	36
7.2 Модифікація даних у віртуальних таблицях.....	36
7.3 Видалення віртуальної таблиці.....	36
8. ЗАХИЩЕНІ БАЗИ ДАНИХ В СУБД FIREBIRD V3.0	37

8.1 Важливість використання засобів захисту баз даних (БД).....	37
8.2 Підходи щодо реалізації захисту баз даних	37
8.3 Способи та архітектура рішень СУБД Firebird v3.0	40
8.4 Аналіз результатів проведення експеримента використання СУБД Firebird v3.0	42
ЛАБОРАТОРНІ РОБОТИ.....	46
Лабораторна робота № 1. Створення бази даних у середовищі MS EXCEL	46
Лабораторна робота № 2. Встановлення СУБД MySQL та СУБД PostgreSQL	65
Лабораторна робота № 3. Візуальне проектування бази даних в MySQL Workbench та в PostgreSQL	93
Лабораторна робота № 4. Додавання даних у базу даних засобами СУБД MySQL та СУБД PostgreSQL	112
Лабораторна робота № 5. Відпрацювання пошуку даних у середовищі системи управління базами даних PostgreSQL	131
Лабораторна робота № 6. Реалізація команд запитів SQL, обмежень та перевірки за допомогою SQL Shell (консолі)	141
Лабораторна робота № 7. Встановлення первинних ключів за допомогою SQL Shell (консолі).....	155
Лабораторна робота № 8. Встановлення зовнішніх ключів засобами MySQL та PostgreSQL.....	158
Лабораторна робота № 9. Реалізація внутрішнього об'єднання за допомогою SQL Shell (консолі).....	170
Лабораторна робота № 10. Розгортання СУБД Firebird 3.0 з підтримкою шифрування даних засобами, розробленими компанією «Сайфер»	173
ЗАВДАННЯ ДЛЯ ВИКОНАННЯ ПІДСУМКОВОЇ РОБОТИ	188
РЕКОМЕНДОВАНА ЛІТЕРАТУРА	220

ПЕРЕДМОВА

“Організація та безпека баз даних” входить до спеціальної (фахової) підготовки бакалавра галузі знань 12 – Інформаційні технології, спеціальності 125 – Кібербезпека та захист інформації, освітньої програми “Кібербезпека”.

Метою викладання навчальної дисципліни “Організація та безпека баз даних” є сформувати у слухачів знання, вміння та навички, необхідні для ефективного використання засобів сучасних інформаційних систем (Систем управління базами даних) у своїй майбутній професійній діяльності, формування у слухачів знань, вмінь та навичок з проєктування, розробки баз даних, використання сучасних мов запитів до баз даних, методів оптимізації, які застосовуються в процесі їх експлуатації, а також підвищення рівня теоретичних знань про основи баз даних.

В програмі навчальної дисципліни “Організація та безпека баз даних” велика увага приділяється теорії реляційних баз даних. При вивченні теорії реляційних баз даних неможливо обійтися без матеріалу, який відноситься до мови запитів SQL. Саме мові запитів SQL присвячено дану методичну розробку, так як на сьогодні інтерфейси, засновані на SQL, підтримуються майже у всіх існуючих системах управління базами даних (СУБД), навіть таких, які не розроблялися як реляційні системи. У даному навчальному посібнику розглянуто формування баз даних в додатку MS Excel, типи даних, допустимі в мові SQL і в SQL-орієнтованих базах даних, а також мовні засоби визначення, маніпулювання, управління даними та транзакціями. Для кращого розуміння студентами теоретичного матеріалу, виклад супроводжується прикладами запитів. Для контролю якості знань студентів в кінці кожної лабораторної роботи надані запитання.

У посібнику детально викладено методичний матеріал даної дисципліни. Він складається з теоретичної частини, де викладено лекційний матеріал, та лабораторного практикуму.

У теоретичній частині розглянуті питання, пов’язані із особливостями формування реляційної моделі даних, формуванням SQL-запиту на вибірку даних, створення бази даних SQL, групуванню та сортуванню даних, використанню підзапитів, правилам маніпуляції з таблицями бази даних та модифікації даних, а також захищені бази даних в СУБД Firebird.

Лабораторний практикум призначений для відображення основних підходів зі створення бази даних в додатку MS Excel, демонстрації виконання

типових операцій із базами даних засобами СУБД MySQL та PostgreSQL, а також розгортанню СУБД Firebird з підтримкою шифрування.

Після засвоєння теоретичного та практичного матеріалу передбачено його закріплення через виконання самостійних завдань за варіантами з наданням необхідного для цього складу тренувальних таблиць баз даних.

Навчальна дисципліна викладається на четвертому курсі бакалаврата за освітньою програмою “Кібербезпека”.

Вона складається з 32 лекційних годин, де студенти можуть опанувати теоретичну інформацію, та лабораторного практикуму (16 годин), що складається з десяти робіт різного рівня складності.

На лабораторний практикум припадає щонайменше 50 балів, які студент повинен набрати, комбінуючи в довільний спосіб кількість виконаних лабораторних робіт.

Посібник призначений для студентів денної та заочної форми навчання.

ЛЕКЦІЙНИЙ МАТЕРІАЛ

1. РЕЛЯЦІЙНА МОДЕЛЬ ДАНИХ

1.1 Структурна частина реляційної бази даних

RDBMS означає Relational Database Management System. Вона є основою для SQL і для всіх сучасних систем баз даних, таких як MS SQL Server, IBM DB2, Oracle, MySQL і PostgreSQL.

Дані в RDBMS зберігаються в об'єктах бази даних, які називаються таблицями. Таблиця – це сукупність пов'язаних записів даних, яка складається зі стовпців і рядків. Кожна таблиця розбивається на менші сутності, які називаються полями. Поля в таблиці «Клієнти» складаються з ідентифікатора клієнта, імені клієнта, імені контакту, адреси, міста, поштового індексу та країни. Поле – це стовпець у таблиці, призначений для збереження конкретної інформації про кожен запис у таблиці.

Запис, який також називають рядком, – це кожен окремий запис, який існує в таблиці. Наприклад, у наведеній вище таблиці "Клієнти" є 91 запис. Запис – це горизонтальна сутність у таблиці.

Стовпець – це вертикальна сутність у таблиці, яка містить усю інформацію, пов'язану з певним полем у таблиці.

1.2 Стандарти SQL

SQL є стандартною мовою для доступу до баз даних і роботи з ними:

SQL означає мову структурованих запитів;

SQL дозволяє отримувати доступ до баз даних і маніпулювати ними;

SQL став стандартом Американського національного інституту стандартів (ANSI) у 1986 році та Міжнародної організації зі стандартизації (ISO) у 1987 році.

Можливості SQL полягають в тому, що:

SQL може виконувати запити до бази даних;

SQL може отримувати дані з бази даних;

SQL може вставляти записи в базу даних;

SQL може оновлювати записи в базі даних;

SQL може видаляти записи з бази даних;

SQL може створювати нові бази даних;

SQL може створювати нові таблиці в базі даних;

SQL може створювати збережені процедури в базі даних;

SQL може створювати представлення в базі даних;

SQL може встановлювати дозволи для таблиць, процедур і представлень;

SQL є стандартом ANSI/ISO, але існують різні версії мови SQL. Однак, щоб бути сумісними зі стандартом ANSI, усі вони підтримують принаймні основні команди (такі як **SELECT**, **UPDATE**, **DELETE**, **INSERT**, **WHERE**) подібним чином.

Примітка. Більшість програм баз даних SQL також мають свої власні розширення на додаток до стандарту SQL!

1.3 Типи даних в SQL. Перетворення типів

Дані – це сукупна інформація, що зберігається в базі даних у вигляді одного з декількох різних типів. За допомогою типів даних встановлюються основні правила для даних, що містяться в конкретному стовпці таблиці, у тому числі розмір пам'яті, що виділяється для них. Основні типи даних – символічні, числові, дата та час.

Символьні дані. Складаються з послідовності символів, що входять у визначений набір символів створеної БД. Найчастіше використовуються набори символів ASCII, UTF, тощо. SQL визначає два первинних символічних типи: із строго заданою кількістю символів (character) та змінною кількістю символів (character varying).

Для визначення даних символічного типу використовується наступний синтаксис:

CHARACTER (n) або CHAR (n) (рядок фіксованої довжини), де n – максимальна кількість символів, що містяться в рядку.

Якщо (n) не зазначено, то передбачається, що рядок складається з одного символу.

Якщо у стовпець типу CHARACTER (n) вводиться $m < n$ символів, то позиції що залишилися заповнюються пробілами.

CHARACTER VARYING (n) АБО VARCHAR (n) (рядок змінної довжини) застосовується тоді, коли дані, що вводяться мають різну довжину і небажано доповнювати їх пробілами. При цьому зберігається тільки та кількість символів, яку ввів користувач. В даному випадку задання максимальної кількості символів обов'язкове.

CHARACTER і CHARACTER VARYING можуть брати участь в одних і тих же стрічкових операціях. Крім цього, у конкретних реалізаціях SQL існують типи для збереження текстових даних “необмеженої” (визначається

архітектурою ЕОМ та особливістю ОС) довжини, наприклад MEMO чи TEXT.

Бітовий тип даних. Використовується для визначення бітових рядків, тобто послідовності двійкових цифр (бітів), кожна з яких може мати значення 0 або 1. Дані бітового типу визначаються за допомогою наступного синтаксису: BIT [VARYING] ()

Числові типи даних. Дані числового типу визначаються точністю і довжиною дробової частини. Точність задає спільну кількість значущих десяткових цифр числа, в яку входить довжина як цілої частки, так і дробової, але без врахування самої десяткової крапки. Масштаб вказує кількість дробових десяткових розрядів числа.

Синтаксис: SMALLINT { INTEGER | INT } BIGINT { NUMERIC | DECIMAL | DEC } [(L)]

Типи NUMERIC і DECIMAL призначені для зберігання чисел в десятковому синтаксисі. За замовчуванням довжина дробової частки дорівнює нулю, а точність, що приймається за умовчанням, залежить від версії SQL.

Тип INTEGER (INT) використовується для зберігання великих додатних або від'ємних цілих чисел.

Тип SMALLINT – для зберігання невеликих додатних або від'ємних цілих чисел. В цьому випадку використання зовнішньої пам'яті істотно зменшується. Дійсні числа або числа з плаваючою крапкою подаються за допомогою мантиси, помноженої на певний степінь десятки (порядку), наприклад: 10E3, +5.2E6, -0.2E-4. Для визначення даних дійсного типу використовується Синтаксис: { FLOAT | REAL } [(N)] DOUBLE PRECISION. Параметр точність задає кількість значущих цифр мантиси. Точність типів REAL і DOUBLE PRECISION залежить від конкретної реалізації.

Тип даних “дата/час”. Даний тип використовується для визначення часу з деякою встановленою точністю. Стандарт SQL підтримує наступний синтаксис: { DATE | TIME | TIMESTAMP } [[WITH TIME ZONE]

Тип даних DATE використовується для зберігання календарних дат, що включають поля YEAR (рік), MONTH (місяць) і DAY (день).

Тип даних TIME – для зберігання відміток часу, що включають поля HOUR (години), MINUTE (хвилини) і SECOND (секунди).

Тип даних TIMESTAMP – для спільного зберігання дати і часу. Параметр задає кількість дробових десяткових знаків, що визначають точність збереження значення в полі SECOND. Якщо цей параметр

опускається, то за замовчуванням його значення, для стовпців типу TIME, встановлюється рівним нулю (тобто зберігаються цілі секунди), тоді як для полів типу TIMESTAMP він встановлюється рівним 6-и (тобто відмітки часу зберігаються з точністю до мілісекунд). Наявність ключової фрази WITH TIME ZONE визначає використання полів TIMEZONE HOUR і TIMEZONE MINUTE, тим самим задаються година і хвилини зміщення зонального часу по відношенню до універсального координатного часу (час за Гринвічем). Дані типу INTERVAL використовуються для представлення періодів часу.

1.4 SQL-операції

Більшість дій, які потрібно виконати з базою даних, виконуються за допомогою операторів SQL.

Інструкції SQL складаються з ключових слів, які легко зрозуміти.

Таблиці бази даних

База даних найчастіше містить одну або декілька таблиць. Кожна таблиця ідентифікується назвою (наприклад, «Клієнти» або «Замовлення») і містить записи (рядки) з даними.

Примітка. Ключові слова SQL НЕ чутливі до регістру: selectme same, що SELECT, тому всі ключові слова SQL у верхньому регістрі.

Крапка з комою після операторів SQL

Деякі системи баз даних вимагають крапки з комою в кінці кожного оператора SQL. Крапка з комою – це стандартний спосіб відокремлення кожного оператора SQL в системах баз даних, які дозволяють виконувати кілька операторів SQL під час одного виклику до сервера. Таким чином, крапка з комою буде ставитись в кінці кожного оператора SQL.

Деякими з найважливіших команд SQL є наступні:

SELECT – витягує дані з бази даних;

UPDATE – оновлює дані в базі даних;

DELETE – видаляє дані з бази даних;

INSERT INTO – вносить нові дані в базу даних;

CREATE DATABASE – створює нову базу даних;

ALTER DATABASE – змінює базу даних;

CREATE TABLE – створює нову таблицю;

ALTER TABLE – змінює таблицю;

DROP TABLE – видаляє таблицю;

CREATE INDEX – створює індекс (ключ пошуку);

DROP INDEX – видаляє індекс.

2. ФОРМУВАННЯ SQL-ЗАПИТУ НА ВИБІРКУ ДАНИХ

2.1 Визначення SQL-запиту

Оператор SQL SELECT

Інструкція **SELECT** використовується для вибору даних із бази даних.

Синтаксис:

SELECT column1, column2, ...

FROM table_name;

де column1, column2, ... – це *назви полів* таблиці, з якої потрібно вибрати дані.

Table_name представляє назву таблиці, з якої потрібно вибрати дані.

Обрати Усі стовпці

Якщо необхідно повернути всі стовпці, не вказуючи ім'я кожного стовпця, можна використати **SELECT ***

Синтаксис:

SELECT*FROM <Назва_таблиці>;

Оператор SQL WHERE

Команда **WHERE** використовується для фільтрації записів, але для вилучення лише тих записів, які відповідають певній умові.

Синтаксис:

SELECT column1, column2, ...

FROM table_name

WHERE condition;

Примітка. Команда WHERE використовується не лише в SELECT-операторах, вона також використовується в UPDATE, DELETE, тощо!

Текстові поля проти числових полів

SQL вимагає одинарних лапок навколо текстових значень (більшість систем баз даних також допускають подвійні лапки). Однак числові поля не слід брати в лапки.

Вираз SQL ORDER BY

Ключове **ORDER BY** слово використовується для сортування набору результатів за порядком зростання або спадання.

Синтаксис:

SELECT column1, column2, ...

FROM table_name

ORDER BY column1, column2, ... **ASC|DESC**;

DESC

За умовчанням ключове **ORDER BY** слово сортує записи за порядком зростання. Щоб відсортувати записи за порядком спадання, необхідно використовувати ключове слово **DESC**.

Перелік в алфавітному порядку

Для рядкових значень **ORDER BY** ключове слово буде впорядковано в алфавітному порядку

За алфавітом DESC

Щоб відсортувати таблицю в зворотному алфавітному порядку, використовуйте **DESC** ключове слово

Оператор SQL AND

Речення **WHERE** може містити один або кілька **AND** операторів.

Оператор **AND** використовується для фільтрації записів на основі кількох умов.

Синтаксис:

SELECT column1, column2, ...

FROM table_name

WHERE condition1 **AND** condition2 **AND** condition3 ...;

Оператор SQL AND проти оператора SQL OR

Оператор **AND** виводить запис, якщо всі умови **ІСТИННІ**.

Оператор **OR** виводить запис, якщо будь-яка з умов є **ІСТИННОЮ**.

Оператор SQL OR

Речення **WHERE** може містити один або декілька **OR** операторів.

Оператор **OR** використовується для фільтрації записів на основі більш ніж однієї умови

Синтаксис:

SELECT column1, column2, ...

FROM table_name

WHERE condition1 **OR** condition2 **OR** condition3 ...;

Оператор OR проти оператора AND

Оператор **OR** виводить запис, якщо будь-яка з умов є **ІСТИННОЮ**.

Оператор **AND** виводить запис, якщо всі умови **ІСТИННІ**.

Поєднання оператора AND та оператора OR

Можна комбінувати оператори **AND** і **OR**.

Оператор NOT

Оператор **NOT** використовується в поєднанні з іншими операторами для отримання протилежного результату, який також називають негативним результатом.

Синтаксис:

SELECT column1, column2, ...

FROM table_name

WHERE NOT condition

Символ підстановки _

Символ **_** підстановки представляє один символ. Це може бути будь-який символ або число, але кожен з них представляє один і тільки один символ.

Символ підстановки %

Символ **%** підстановки представляє будь-яку кількість символів, навіть нуль символів.

Починається з

Щоб повернути записи, які починаються з певної літери або фрази, додайте **%** в кінці літери або фрази.

*Примітка. Також можна комбінувати будь-яку кількість умов за допомогою операторів **AND** або **OR**.*

Закінчується на

Щоб повернути записи, які закінчуються певною літерою чи фразою, додайте **%** на початок літери чи фрази.

Примітка. Можна комбінувати "починається з" і "закінчується з".

Містить

Щоб повернути записи, які містять певну літеру чи фразу, додайте % до та після літери чи фрази.

Об'єднання символів підстановки

Будь-який символ узагальнення, наприклад % і _, можна використовувати в поєднанні з іншими символами підстановки.

Без символа підстановки

Якщо символ підстановки не вказано, фраза повинна мати точний збіг, щоб повернути результат.

Символи підстановки SQL

Символ підстановки використовується для заміни одного або кількох символів в рядку. З оператором використовуються символи підстановки. Оператор використовується з метою пошуку заданого шаблону в стовпці. LIKE WHERE

Псевдоніми SQL

Псевдоніми SQL використовуються для надання таблиці або стовпцю в таблиці тимчасового імені. Псевдоніми часто використовуються, щоб зробити назви стовпців більш читабельними. Псевдонім існує лише протягом певного запиту. Псевдонім створюється за допомогою AS ключового слова, але AS може бути необов'язковим. Фактично, у більшості мов баз даних можна пропустити ключове слово AS і отримати той самий результат.

Синтаксис:

- коли псевдонім використовується в стовпці:

SELECT column_name **AS** alias_name

FROM table_name;

- коли в таблиці використовується псевдонім:

SELECT column_name(s)

FROM table_name **AS** alias_name;

Інструкція SQL SELECT INTO

Оператор **SELECT INTO** копіює дані з однієї таблиці в нову таблицю.

Синтаксис:

- скопіювати всі стовпці в нову таблицю:

SELECT *

INTO newtable [**IN** externaldb]

FROM oldtable

WHERE condition;

- скопіювати лише деякі стовпці в нову таблицю:

SELECT column1, column2, column3, ...

INTO newtable [IN externaldb]

FROM oldtable

WHERE condition

Нова таблиця буде створена з назвами стовпців і типами, як визначено в старій таблиці. Можна створити нові назви стовпців за допомогою AS.

Інструкція SQL INSERT INTO SELECT

Оператор INSERT INTO SELECT копіює дані з однієї таблиці та вставляє їх в іншу таблицю. Дана інструкція вимагає, щоб типи даних у вихідній і цільовій таблицях збігалися.

Примітка. Існуючі записи в цільовій таблиці не впливають.

Синтаксис:

- скопіювати всі стовпці з однієї таблиці в іншу:

INSERT INTO table2

SELECT * FROM table1

WHERE condition;

- скопіювати лише деякі стовпці з однієї таблиці в іншу:

INSERT INTO table2 (column1, column2, column3, ...)

SELECT column1, column2, column3, ...

FROM table1

WHERE condition;

Оператор SQL CASE

Оператор CASE проходить через умови та повертає значення, коли виконується перша умова (як оператор if-then-else). Отже, коли умова виконується, вона припинить читання та поверне результат. Якщо жодна умова не виконується, повертається значення в ELSE пропозиції.

Якщо частина відсутня ELSE і умови не виконуються, повертається NULL.

Синтаксис:

CASE

WHEN condition1 **THEN** result1

WHEN condition2 **THEN** result2

WHEN conditionN **THEN** resultN

ELSE result

END;

2.2 Оператори SQL

2.2.1 Арифметичні оператори SQL

Оператор	Опис
+	Додати
-	Відняти
*	Помножити
/	Поділити

2.2.2 Побітові оператори SQL

Оператор	Опис
&	Бітове AND
	Бітове OR
^	Бітовий ексклюзив OR

2.2.3 Оператори порівняння SQL

Оператор	Опис
=	Дорівнює
>	Більше ніж
<	Менше ніж
>=	Більше або дорівнює
<=	Менше або дорівнює
<>	Не дорівнює

2.2.4 Складені оператори SQL

Оператор	Опис
+=	Додавання дорівнює
-=	Віднімання рівностей
*=	Помножити рівності
/=	Розділити дорівнює
%=	По модулю дорівнює
&=	Порозрядне І дорівнює
^-=	Порозрядне виключаюче дорівнює
*=	Порозрядне АБО дорівнює

2.2.5 Логічні оператори SQL

Оператор	Опис
ALL	ІСТИНА, якщо всі значення підзапиту відповідають умові
AND	ІСТИНА, якщо всі умови, розділені AND, ІСТИНА
ANY	ІСТИНА, якщо будь-яке зі значень підзапиту відповідає умові
BETWEEN	ІСТИНА, якщо операнд перебуває в діапазоні порівнянь
EXISTS	TRUE, якщо підзапит повертає один або кілька записів
IN	ІСТИНА, якщо операнд дорівнює одному з виразів зі списку
LIKE	TRUE, якщо операнд відповідає шаблону
NOT	Відображає запис, якщо умова (s) є NOT TRUE
OR	ІСТИНА, якщо будь-яка з умов, розділених АБО, є ІСТИНОЮ
SOME	TRUE, якщо будь-яке зі значень підзапиту відповідає умові

2.2.6 Оператори визначення даних

CREATE	створює об'єкт БД (саму базу, таблицю, подання, користувача тощо)
ALTER	змінює об'єкт
DROP	видаляє об'єкт

2.2.7 Оператори маніпуляції даними

SELECT	вибирає дані, що задовольняють заданим умовам
INSERT	додає нові дані
UPDATE	змінює наявні дані
DELETE	видаляє дані

2.2.8 Оператори визначення доступу до даних

GRANT	надає користувачеві (групі) дозволи на певні операції з об'єктом
REVOKE	відкликає раніше видані дозволи
DENY	задає заборону, що має пріоритет над дозволом

2.2.9 Оператори управління транзакціями

COMMIT	застосовує транзакцію
ROLLBACK	відкочує всі зміни, зроблені в контексті поточної транзакції
SAVEPOINT	ділить транзакцію на більш дрібні ділянки

3. БАЗА ДАНИХ SQL

3.1 Створення бази даних

Команда SQL CREATE DATABASE

Цей **CREATE DATABASE** оператор використовується для створення нової бази даних SQL.

Синтаксис:

CREATE DATABASE *databasename*;

Команда SQL DROP DATABASE

Інструкція **DROP DATABASE** використовується для видалення існуючої бази даних SQL.

Синтаксис:

DROP DATABASE *databasename*;

Команда SQL BACKUP DATABASE

Цей **BACKUP DATABASE** оператор використовується в SQL Server для створення повної резервної копії існуючої бази даних SQL.

Синтаксис:

BACKUP DATABASE *databasename* **TO DISK** = '*filepath*';

Команда SQL BACKUP WITH DIFFERENTIAL

Диференціальне резервне копіювання створює резервні копії лише тих частин бази даних, які змінилися після останнього повного резервного копіювання бази даних.

Синтаксис:

BACKUP DATABASE *databasename*
TO DISK = '*filepath*'
WITH DIFFERENTIAL;

3.2 Усунення надлишковості вибраних даних.

Команда SQL SELECT DISTINCT

Оператор **SELECT DISTINCT** використовується для повернення лише різних (різних) значень. У середині таблиці стовпець часто містить багато повторюваних значень; іноді вам потрібно лише перелічити різні (різні) значення.

Синтаксис:

```
SELECT DISTINCT column1, column2, ...  
FROM table_name;
```

3.3 Уточнення запиту за допомогою предикатів

Фільтрування рядків в SQL-запитах

Оператори в реченні WHERE. Для фільтрації пошуку можна використовувати інші оператори, крім оператора.

У запиті можна використовувати такі оператори **WHERE**:

Оператор	Опис
=	Рівні
>	Більше ніж
<	Менше ніж
>=	Більше або дорівнює
<=	Менше або дорівнює
<>	Не рівні. Примітка. У деяких версіях SQL цей оператор може бути записаний як !=
BETWEEN	Між певним діапазоном
LIKE	Пошук візерунка
IN	Щоб указати кілька можливих значень для стовпця

3.4 Спеціальні SQL-предикати. Належність до множини

Оператор SQL IN

Оператор **IN** дозволяє вказати кілька значень у **WHERE** реченні.

Оператор **IN** є скороченням кількох **OR** умов.

Синтаксис:

```
SELECT column_name(s)  
FROM table_name  
WHERE column_name IN (value1, value2, ...);
```

Оператор SQL NOT IN

Використовуючи **NOT** ключове слово перед **IN** оператором, повертаються всі записи, які **НЕ** є жодними із значень в списку.

Оператор SQL IN (WHERE)

Також можна використовувати **IN** з підзапитом у **WHERE** реченні. За допомогою підзапиту можна повернути всі записи з основного запиту, які присутні в результаті підзапиту.

3.5 Предикат діапазону

Оператор SQL BETWEEN

Оператор **BETWEEN** вибирає значення в заданому діапазоні. Значеннями можуть бути числа, текст або дати. Оператор **BETWEEN** включає: початкове та кінцеве значення включено.

Синтаксис:

SELECT column_name(s)

FROM table_name

WHERE column_name **BETWEEN** value1 **AND** value2;

Оператор SQL NOT BETWEEN

Щоб відобразити об'єкти поза діапазоном, необхідно використовувати **NOT BETWEEN**.

3.6 Предикат шаблону

Оператор SQL LIKE та NOT LIKE

Оператор **LIKE** використовується в **WHERE** пропозиції для пошуку заданого шаблону в стовпці.

У поєднанні з оператором часто використовуються два символи підстановки **LIKE**:

- знак відсотка % означає нуль, один або декілька символів;
- знак підкреслення _ означає один символ.

Синтаксис:

SELECT column1, column2, ...

FROM table_name

WHERE columnN **LIKE** pattern;

3.7 Перевірка на значення NULL

Поле зі значенням **NULL** є полем без значення. Якщо поле в таблиці є необов'язковим, можна вставити новий запис або оновити запис, не додаючи значення до цього поля. Тоді поле буде збережено зі значенням **NULL**.

*Примітка. Значення **NULL** відрізняється від нульового значення або поля, яке містить пробіли. Поле зі значенням **NULL** – це поле, яке було залишено порожнім під час створення запису!*

Неможливо перевірити значення **NULL** за допомогою операторів порівняння, таких як **=**, **<** або **<>**. Замість цього доведеться використовувати оператори **IS NULL** and **IS NOT NULL**.

Синтаксис:

```
SELECT column_names  
FROM table_name  
WHERE column_name IS NULL;  
IS NOT NULL Синтаксис  
SELECT column_names  
FROM table_name  
WHERE column_name IS NOT NULL;
```

Оператор IS NULL

Оператор **IS NULL** використовується для перевірки порожніх значень (значення **NULL**).

Примітка. Завжди використовуйте **IS NULL** для пошуку значень **NULL**.

Оператор IS NOT NULL

Оператор **IS NOT NULL** використовується для перевірки непорожніх значень (**NE NULL**).

4. ГРУПУВАННЯ ТА СОРТУВАННЯ ДАНИХ. ВИКОРИСТАННЯ ПІДЗАПИТІВ

4.1 Групування записів за значеннями одного або декількох стовпців

Команда SQL GROUP BY

Інструкція **GROUP BY** групує рядки з однаковими значеннями в підсумкові рядки. Оператор **GROUP BY** часто використовується з агрегатними функціями (**COUNT()**, **MAX()**, **MIN()**, **SUM()**, **AVG()**) для групування набору результатів за одним або кількома стовпцями.

Синтаксис:

```
SELECT column_name(s)
FROM table_name
WHERE condition
GROUP BY column_name(s)
ORDER BY column_name(s);
```

Команда SQL HAVING

Речення **HAVING** було додано до SQL, оскільки **WHERE** ключове слово не можна використовувати з агрегатними функціями.

Синтаксис:

```
SELECT column_name(s)
FROM table_name
WHERE condition
GROUP BY column_name(s)
HAVING condition
ORDER BY column_name(s);
```

4.2 Використання агрегатних функцій

Агрегатні функції SQL

Агрегатна функція – це функція, яка обчислює набір значень і повертає єдине значення. Агрегатні функції часто використовуються з **GROUP BY** умовою оператора **SELECT**. Інструкція **GROUP BY** розбиває набір результатів на групи значень, а агрегатну функцію можна використовувати для повернення окремого значення для кожної групи.

Найпоширенішими агрегатними функціями SQL є:

MIN() – повертає найменше значення у вибраному стовпці;

MAX() – повертає найбільше значення у вибраному стовпці;

COUNT() – повертає кількість рядків у наборі;

SUM() – повертає загальну суму числового стовпця;

AVG() – повертає середнє значення числового стовпця.

Агрегатні функції ігнорують нульові значення (крім **COUNT()**).

Функції SQL MIN() і MAX().

Функція **MIN()** повертає найменше значення вибраного стовпця.

Функція **MAX()** повертає найбільше значення вибраного стовпця.

Синтаксис:

SELECT MIN(column_name)

FROM table_name

WHERE condition;

SELECT MAX(column_name)

FROM table_name

WHERE condition

Встановити назву стовпця (псевдонім)

Якщо використовують **MIN()** або **MAX()**, повернутий стовпець не матиме описової назви. Щоб дати стовпцю описову назву, використовуйте **AS** ключове слово

Функція SQL COUNT().

Функція **COUNT()** повертає кількість рядків, які відповідають заданому критерію.

Синтаксис:

SELECT COUNT(column_name)

FROM table_name

WHERE condition;

Вказати стовпець

Можна вказати назву стовпця замість символу зірочки (*). Якщо вказати назву стовпця замість (*), значення NULL не будуть зараховані.

Додати інструкцію WHERE

Можна додати **WHERE** пункт, щоб визначити умови.

Ігнорувати дублікати

Можна ігнорувати дублікати, використовуючи **DISTINCT** у **COUNT()** функції. Якщо **DISTINCT** вказано, рядки з однаковим значенням для вказаного стовпця вважатимуться одним.

Функція SQL SUM().

Функція **SUM()** повертає загальну суму числового стовпця.

Синтаксис:

SELECT SUM(column_name)

FROM table_name

WHERE condition;

Функція SQL AVG().

Функція **AVG()** повертає середнє значення числового стовпця.

Примітка. Значення NULL ігноруються.

Синтаксис:

SELECT AVG(column_name)

FROM table_name

WHERE condition;

4.3 Використанням кванторів

Оператори SQL ANY і ALL

Оператори **ANY** and **ALL** дозволяють виконувати порівняння між значеннями одного стовпця та діапазоном інших значень.

Оператор SQL ANY

Оператор **ANY**:

у результаті повертає логічне значення;

повертає **TRUE**, якщо **БУДЬ-ЯКЕ** зі значень підзапиту відповідає умові.

ANY означає, що умова буде істинною, якщо операція буде істинною для будь-якого зі значень у діапазоні.

Синтаксис:

SELECT column_name(s)

FROM table_name

WHERE column_name operator **ANY**

(**SELECT** column_name

FROM table_name

WHERE condition);

Примітка: оператор має бути стандартним оператором порівняння (=, <>, !=, >, >=, < або <=) .

Оператор SQL ALL

Оператор **ALL**:

- у результаті повертає логічне значення;
- повертає **TRUE**, якщо **ВСІ** значення підзапиту відповідають умові;
- використовується з **SELECT** і **WHERE** операторами **HAVING**.

ALL означає, що умова буде істинною, лише якщо операція буде істинною для всіх значень у діапазоні.

Синтаксис з **SELECT**:

```
SELECT ALL column_name(s)  
FROM table_name  
WHERE condition;
```

Синтаксис з **WHERE** або **HAVING**:

```
SELECT column_name(s)  
FROM table_name  
WHERE column_name operator ALL  
(SELECT column_name  
FROM table_name  
WHERE condition);
```

Примітка: оператор має бути стандартним оператором порівняння (=, <>, !=, >, >=, < або <=).

4.4 Використання предиката EXISTS

Оператор SQL EXISTS

Оператор **EXISTS** використовується для перевірки існування будь-якого запису в підзапиті. Оператор **EXISTS** повертає **TRUE**, якщо підзапит повертає один або більше записів.

Синтаксис:

```
SELECT column_name(s)  
FROM table_name  
WHERE EXISTS  
(SELECT column_name FROM table_name WHERE condition);
```

5. ФОРМУВАННЯ СКЛАДНИХ SQL-ЗАПИТІВ

5.1 Використання теоретико-множинних операцій – формування складних SQL-запитів з використанням оператора INNER JOIN

SQL JOIN

Речення **JOIN** використовується для об'єднання рядків з двох або більше таблиць на основі пов'язаного стовпця між ними.

ВНУТРІШНЄ З'ЄДНАННЯ

INNER JOIN обирає записи, які мають відповідні значення в обох таблицях (рис. 1).

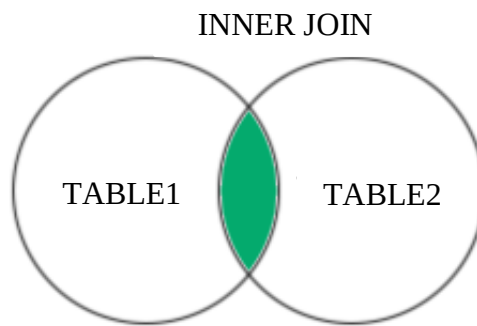


Рис. 1. Графічне відображення виконання операції **INNER JOIN**

Синтаксис:

```
SELECT column_name(s)  
FROM table1  
INNER JOIN table2  
ON table1.column_name = table2.column_name;
```

JOIN або INNER JOIN

JOIN і **INNER JOIN** поверне той самий результат. **INNER** є типовим типом об'єднання для **JOIN**, тому, коли ви пишете, **JOIN** аналізатор фактично записує **INNER JOIN**.

5.2 Використання теоретико-множинних операцій – формування складних SQL-запитів з використанням оператора LEFT JOIN

SQL LEFT JOIN

Ключове **LEFT JOIN** слово повертає всі записи з лівої таблиці (TABLE1) і відповідні записи з правої таблиці (TABLE2) (рис. 2). Результатом буде 0 записів з правого боку, якщо немає відповідності.

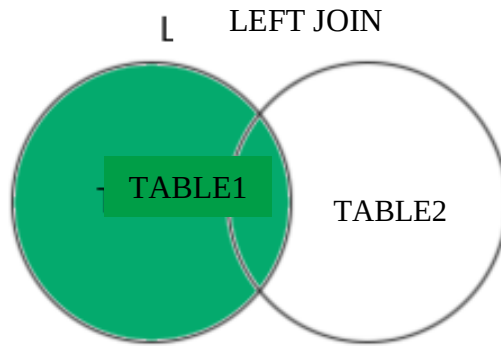


Рис. 2. Графічне відображення виконання операції LEFT JOIN

Синтаксис:

```
SELECT column_name(s)
FROM table1
LEFT JOIN table2
ON table1.column_name = table2.column_name;
```

Примітка. У деяких базах даних LEFT JOIN називається LEFT OUTER JOIN.

5.3 Використання теоретико-множинних операцій – формування складних SQL-запитів з використанням оператора RIGHT JOIN

SQL RIGHT JOIN

Ключове **RIGHT JOIN** слово повертає всі записи з правої таблиці (TABLE2) і відповідні записи з лівої таблиці (TABLE1) (рис. 3). Результатом буде 0 записів зліва, якщо немає відповідності.

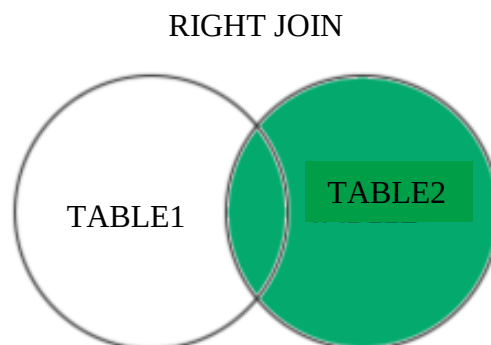


Рис. 3. Графічне відображення виконання операції RIGHT JOIN

Синтаксис:

```
SELECT column_name(s)  
FROM table1  
RIGHT JOIN table2  
ON table1.column_name = table2.column_name;
```

Примітка. У деяких базах даних RIGHT JOIN називається RIGHT OUTER JOIN.

5.4 Використання теоретико-множинних операцій – формування складних SQL-запитів з використанням оператора FULL OUTER JOIN

SQL FULL OUTER JOIN

Ключове **FULL OUTER JOIN** слово повертає всі записи, якщо є відповідність у лівих (TABLE1) або правих (TABLE2) записах таблиці (рис. 4).

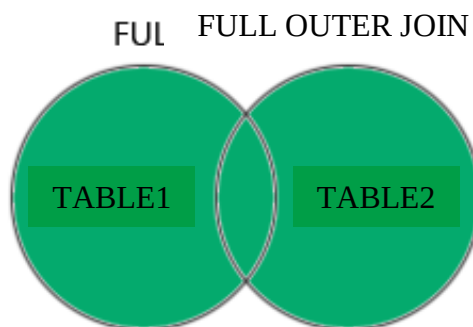


Рис. 4. Графічне відображення виконання операції FULL OUTER JOIN

Примітка. **FULL OUTER JOIN** і **FULL JOIN** однакові.

Синтаксис:

```
SELECT column_name(s)  
FROM table1  
FULL OUTER JOIN table2  
ON table1.column_name = table2.column_name  
WHERE condition;
```

SQL SELF JOIN

Самооб'єднання є звичайним об'єднанням, але таблиця об'єднується сама з собою.

Синтаксис:

```
SELECT column_name(s)  
FROM table1 T1, table1 T2  
WHERE condition
```

T1 і T2 є різними псевдонімами для однієї таблиці.

5.5 Використання теоретико-множинних операцій – формування складних SQL-запитів з використанням оператора **UNION** та **UNION ALL**

SQL UNION

Оператор **UNION** використовується для поєднання набору результатів двох або більше **SELECT** операторів. Кожен **SELECT** оператор усередині **UNION** повинен мати однакову кількість стовпців. Стовпці також повинні мати подібні типи даних. Стовпці в кожній **SELECT** заяві також мають бути в однаковому порядку.

Синтаксис:

```
SELECT column_name(s) FROM table1  
UNION  
SELECT column_name(s) FROM table2;
```

SQL UNION ALL

UNION за замовчуванням оператор вибирає лише різні значення. Щоб дозволити повторювані значення, використовуйте **UNION ALL**.

Синтаксис:

```
SELECT column_name(s) FROM table1  
UNION ALL  
SELECT column_name(s) FROM table2;
```

*Примітка. Назви стовпців у наборі результатів зазвичай збігаються з назвами стовпців у першому **SELECT** операторі.*

6. ПРАВИЛА МАНІПУЛЯЦІЇ З ТАБЛИЦЯМИ ТА МОДИФІКАЦІЇ ДАНИХ

6.1 Створення таблиць

SQL CREATE TABLE

CREATE TABLE використовується для створення нової таблиці в базі даних.

Синтаксис:

```
CREATE TABLE table_name (  
    column1 datatype,  
    column2 datatype,  
    column3 datatype,  
);
```

Параметри стовпців задають імена стовпців таблиці. Параметр *datatype* визначає тип даних, які може містити стовпець (наприклад, *varchar*, *integer*, *date* тощо).

6.2 Обмеження на дані

Обмеження SQL використовуються для визначення правил для даних у таблиці. Обмеження можна вказати під час створення таблиці за допомогою **CREATE TABLE** оператора або після створення таблиці за допомогою **ALTER TABLE** оператора.

Синтаксис:

```
CREATE TABLE table_name (  
    column1 datatype constraint,  
    column2 datatype constraint,  
    column3 datatype constraint,  
    ....);
```

Обмеження використовуються для обмеження типу даних, які можуть входити в таблицю. Це забезпечує точність і достовірність даних у таблиці. Якщо існує будь-яке порушення між обмеженням і дією з даними, дія переривається. Обмеження можуть бути на рівні стовпця або таблиці.

Обмеження рівня стовпця застосовуються до стовпця, а обмеження рівня таблиці застосовуються до всієї таблиці.

У SQL зазвичай використовуються такі обмеження:

NOT NULL – гарантує, що стовпець не може мати значення NULL;

UNIQUE – гарантує, що всі значення в стовпці відрізняються;

PRIMARY KEY – поєднання **NOT NULL** і **UNIQUE**. Унікально визначає кожен рядок у таблиці;

FOREIGN KEY – запобігає діям, які можуть зруйнувати зв'язки між таблицями;

CHECK – переконується, що значення в стовпці задовольняють певну умову;

DEFAULT – встановлює значення за замовчуванням для стовпця, якщо значення не вказано;

CREATE INDEX – використовується для створення та отримання даних з бази даних дуже швидко.

Обмеження SQL NOT NULL

За замовчуванням стовпець може містити значення NULL. Обмеження **NOT NULL** змушує стовпець НЕ приймати значення NULL. Це означає, що поле завжди містить значення, що означає, що ви не можете вставити новий запис або оновити запис, не додавши значення до цього поля.

Обмеження SQL UNIQUE

Обмеження **UNIQUE** гарантує, що всі значення в стовпці відрізняються. І обмеження **UNIQUE**, **PRIMARY KEY** і забезпечують гарантію унікальності для стовпця або набору стовпців.

Обмеження SQL PRIMARY KEY

Автоматично має **UNIQUE** обмеження. Однак можна мати багато **UNIQUE** обмежень на таблицю, але лише одне **PRIMARY KEY** обмеження на таблицю.

Обмеження **PRIMARY KEY** однозначно ідентифікує кожен запис у таблиці. Первинні ключі повинні містити значення **UNIQUE** і не можуть містити значення NULL. Таблиця може мати лише ОДИН первинний ключ; а в таблиці цей первинний ключ може складатися з одного або кількох стовпців (полів).

Обмеження SQL FOREIGN KEY

Обмеження **FOREIGN KEY** використовується для запобігання діям, які руйнують зв'язки між таблицями.

FOREIGN KEY – це поле (або сукупність полів) в одній таблиці, яке посилається на поле **PRIMARY KEY** в іншій таблиці.

Таблиця із зовнішнім ключем є дочірньою таблицею, а таблиця з первинним ключем – батьківською таблицею.

Обмеження SQL CHECK

Обмеження **CHECK** використовується для обмеження діапазону значень, які можна розмістити в стовпці. Якщо визначити **CHECK** обмеження для стовпця, це дозволить лише певні значення для цього стовпця. Якщо ви визначаєте **CHECK** обмеження для таблиці, воно може обмежувати значення в певних стовпцях на основі значень в інших стовпцях рядка.

Обмеження SQL DEFAULT

Обмеження **DEFAULT** використовується для встановлення значення за замовчуванням для стовпця. Значення за замовчуванням буде додано до всіх нових записів, якщо не вказано інше значення.

6.3 Модифікація таблиць

SQL ALTER TABLE

ALTER TABLE використовується для додавання, видалення або зміни стовпців у існуючій таблиці. **ALTER TABLE** також використовується для додавання та видалення різноманітних обмежень на існуючу таблицю.

Для проведення модифікації таблиці через додавання стовпця, необхідно використовувати такий синтаксис:

```
ALTER TABLE table_name  
ADD column_name datatype
```

6.4 Видалення таблиць

Команда SQL DROP TABLE

Щоб повністю видалити таблицю, користуються **DROP TABLE** оператором. **DROP TABLE** використовується для видалення існуючої таблиці в базі даних.

Синтаксис:

```
DROP TABLE table_name;
```

6.5 Внесення даних

SQL INSERT INTO

Використовується для вставки нових записів у таблицю.

Синтаксис:

1) вказати назви стовпців і значення, які потрібно вставити:

INSERT INTO table_name (column1, column2, column3, ...)

VALUES (value1, value2, value3, ...);

2) якщо додаються значення для всіх стовпців таблиці, то потрібно вказувати назви стовпців у SQL-запиті. Однак треба переконатися, що порядок значень відповідає порядку стовпців у таблиці:

INSERT INTO table_name

VALUES (value1, value2, value3, ...);

6.6 Зміна даних

SQL UPDATE

Використовується для зміни наявних записів у таблиці.

Синтаксис:

UPDATE table_name

SET column1 = value1, column2 = value2, ...

WHERE condition;

Примітка. Необхідно бути обережним при оновленні записів в таблиці! Зверніть увагу на використання WHERE у UPDATE. У WHERE вказується які записи потрібно оновити. Якщо опустити WHERE, усі записи в таблиці будуть оновлені!

ОНОВЛЕННЯ кількох записів

WHERE, який визначає, скільки записів буде оновлено.

Примітка. Будьте обережні при оновленні записів. Якщо ви опустите WHERE, ВСІ записи будуть оновлені!

SQL SELECT TOP

Використовується для визначення кількості записів, які потрібно повернути. Він корисний для великих таблиць із тисячами записів. Повернення великої кількості записів може вплинути на продуктивність.

*Примітка. Не всі системи баз даних підтримують **SELECT TOP**. MySQL підтримує **LIMIT** для вибору обмеженої кількості записів.*

Синтаксис:

```
SELECT column_name(s)  
FROM table_name  
WHERE condition  
LIMIT number;
```

6.7 Видалення даних

SQL DELETE

Використовується для видалення існуючих записів у таблиці.

Синтаксис:

```
DELETE FROM table_name WHERE condition.
```

*Примітка. Будьте обережні, видаляючи записи в таблиці! Зверніть увагу на **WHERE** у **DELETE**. У **WHERE** вказується, які записи слід видалити. Якщо опустити **WHERE**, усі записи в таблиці будуть видалені!*

Видалити всі записи

Можна видалити всі рядки в таблиці, не видаляючи таблицю. Це означає, що структура таблиці, атрибути та індекси залишаться недоторканою.

Синтаксис:

```
DELETE FROM table_name.
```

7. ВІРТУАЛЬНІ ТАБЛИЦІ (VIEW – ПРЕДСТАВЛЕННЯ)

7.1 Створення віртуальних таблиць

SQL CREATE VIEW

У SQL представлення – це віртуальна таблиця, заснована на наборі результатів оператора SQL. Подання містить рядки та стовпці, як і справжня таблиця. Поля в представленні – це поля з однієї чи кількох реальних таблиць у базі даних. Можна додати оператори та функції SQL до подання та представити дані так, ніби вони надходять з однієї таблиці.

Синтаксис:

```
CREATE VIEW view_name AS  
SELECT column1, column2, ...  
FROM table_name  
WHERE condition;
```

Примітка. Перегляд завжди відображає актуальні дані! Механізм бази даних відтворює представлення кожного разу, коли користувач запитує його.

7.2 Модифікація даних у віртуальних таблицях

SQL Оновлення представлення

Подання можна оновити за допомогою **CREATE OR REPLACE VIEW**.

Синтаксис:

```
CREATE OR REPLACE VIEW view_name AS  
SELECT column1, column2, ...  
FROM table_name  
WHERE condition;
```

7.3 Видалення віртуальної таблиці

SQL Видалення представлення

Подання видаляється разом із **DROP VIEW** оператором.

Синтаксис:

```
DROP VIEW view_name
```

8. ЗАХИЩЕНІ БАЗИ ДАНИХ В СУБД FIREBIRD V3.0

8.1 Важливість використання засобів захисту баз даних (БД)

У сучасному інформаційному світі, інформація відіграє значну роль у житті людини, суспільства і держави. Зростання розміру накопичуваних і оброблюваних даних порушує питання про їх зберігання та забезпечення конфіденційності. Уже існує чимало технічних рішень і пропозицій для вирішення подібних завдань. Серед них є і системи управління базами даних (СУБД), які підтримують шифрування збережених даних.

У квітні 2016 року вийшла нова третя версія СУБД Firebird. З нововведень з'явилося і чимало механізмів захисту даних, що зберігаються і передаються. Там є і захист каналу передачі даних, є управління користувачами, а також є шифрування самої БД, яке реалізовано як прозоре шифрування на рівні сторінок даних. Реалізується це все за допомогою написання спеціальних розширень для Firebird. C++ Firebird API.

Види даних, для яких необхідно забезпечити їх конфіденційність під час зберігання в БД:

- 1) персональні дані;
- 2) чутливі дані:
 - медичні;
 - банківські;
 - інші.

Нормативними вимогами для захисту даних є:

- 1) КСЗІ;
- 2) ISO/IEC 27001:2013;
- 3) PCI-DSS;
- 4) HIPAA;
- 5) FERPA.

Ці вимоги визначають необхідність шифрування персональних даних.

8.2 Підходи щодо реалізації захисту баз даних

Існуючі підходи до захисту інформації у вигляді процедури шифрування такі:

- 1) шифрування з'єднання;
- 2) шифрування файлів БД:

- шифрування всього розділу (рівень ФС);
- шифрування вказаних файлів (рівень ФС);
- використання спеціальних інструментів (BitLocker, TrueCrypt, VeraCrypt та інші);

3) шифрування даних у БД (TDE):

- прозоре шифрування простору таблиць, посторінково;

4) шифрування полів у БД (AL):

- на основі паролів в SQL.

Наявними рішеннями у світі СУБД є:

1) Oracle DB Enterprise (TDE, AL):

- ключ у файлі;
- ключ на пристрої;

2) Microsoft SQL Server Enterprise (TDE, AL):

- ключ у файлі;
- ключ на пристрої;

3) MongoDB Enterprise (TDE, AL):

- ключ у файлі;
- ключ на пристрої;
- заснований на OpenSSL;

4) MySQL Enterprise (TDE, AL):

- ключ у файлі;
- ключ на пристрої;

5) MySQL/MariaDB/Percona (AL):

- пароль у SQL/DB;

6) PostgreSQL (AL):

- використовується pgcrypt (plugin загального призначення);
- пароль у SQL/DB;

7) SQLite.

До нових рішень на сьогодні відносять СУБД Firebird v3.0 з такими характеристиками:

- промислова СУБД;
- безоплатна;
- відкритий вхідний код;
- великі обсяги даних;
- висока стабільність;

- підтримка шифрування БД;
- підтримка шифрування каналу;
- особиста система управління користувачами;
- робота офлайн із шифрованими БД.

Деталіми реалізації на Firebird C++API є такі розширення:

- 1) розширення-зберігач секретного ключа (KeyHolder);
- 2) розширення шифрування даних (DbCrypt);

При цьому реалізовується зашифровування/розшифровування БД за допомогою запиту SQL:

`ALTER DATABASE ENCRYPT WITH "plugin_name" KEY "key_name".`

- 3) протокол безпечного передавання ключа:

- DbCrypt plugin → KeyHolder plugin: прошу передати мені ключ;
- KeyHolder plugin: шифрує ключ (з урахуванням даних від DbCrypt);
- KeyHolder plugin → DbCrypt plugin: зашифрований ключ;
- DbCrypt plugin: розшифровує і перевіряє ключ;
- `ALTER DATABASE DECRYPT`: готовий до роботи;

- 4) шифрування у фоновому службовому потоці;

- 5) можливість управління ключами;

У такий спосіб можуть бути зашифровані сторінки даних, blob та індексів (окрім заголовка сторінки) - зашифровані, включно із системними таблицями, а допоміжні сторінки не зашифровані, оскільки не містять конфіденційної інформації;

- 6) перевірка коректності ключа проводиться за допомогою порівняння хеша фіксованих даних зі зразком, що зберігається у заголовку;

- 7) особливо важливі дані захищені додатковою MAC.

Під час використання цього додатка пропонуються такі засоби, розроблені для сторонніх розробників, які дають змогу захистити використовувану їхнім додатком БД від несанкціонованого доступу:

- Plugin для доступу до ключа;
- Plugin для шифрування БД;
- модуль активації з описом інтерфейсу для використання в програмному забезпеченні, що розробляється;
- підтримка мережевого та Embedded режиму.

8.3 Способи та архітектура рішень СУБД Firebird v3.0

Варіантами використання є попередньо встановлені дані (підтримка шифрування БД, призначених для передавання та подальшого використання іншою особою; доступ можливий тільки за ключем) і впровадження в готове рішення (без особливих змін в основний застосунок можна забезпечити конфіденційність даних у БД, не витрачати час на розбір підходів та алгоритмів забезпечення конфіденційності, не витрачати час на знайомство з C++ API Firebird і написання плагінів).

Архітектура рішення цього додатка представлена на рис. 5.

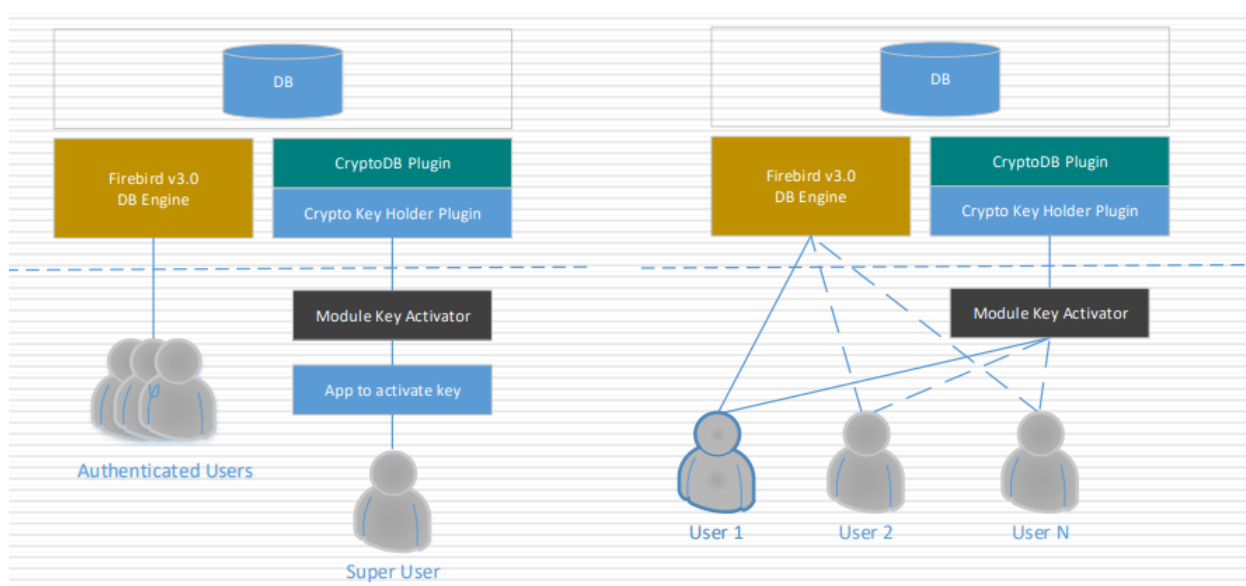


Рис. 5. Архітектура Firebird C++API

Так, у готове рішення системи шифрування БД для СУБД Firebird 3 вХОДЯТЬ:

- Plugin для доступу до ключа;
- Plugin для шифрування БД;
- сервер активації ключів;
- клієнт генерації контейнерів ключів та управління ними.

До завдань, що вирішуються за допомогою СУБД Firebird 3 відносяться:

1) безпечне зберігання секретного ключа БД:

- зберігається в зашифрованому вигляді у файловому контейнері;
- є можливість зберігати в контейнері тільки пароль до захищеного носія;

- активація доступна тільки двом відповідальним особам;
- кількість відповідальних осіб не обмежена;
- число використовуваних ключів системою не обмежене;

2) шифрування даних:

- після того, як ключ було активовано, система працює автоматично і не потребує участі людини;
- число зашифрованих БД системою не обмежене;

3) безпечне використання секретного ключа БД:

- доступ до секретного ключа забезпечує три елементи системи;
- обмін інформацією між елементами забезпечується шифруванням на сесійних ключах;
- об'єкти доступу до ключа створюються динамічно і залежно від налаштувань живуть не більше організованої сесії;
- тільки під час запуску процесу шифрування ключ перетворюється і готовий до використання;

4) застосування системи:

- є можливість зберігати необмежену кількість контейнерів секретних ключів і при цьому шифрувати в різних режимах необмежену кількість баз даних;
- обслуговуючому персоналу сервера, на якому розгорнуто систему шифрування БД, недоступні дані, що зберігаються;
- можна віддавати на зберігання БД із чутливими даними в аутсорсинг;
- захист БД від фізичного викрадення;
- може використовуватися в моделі SaaS.

Для шифрування даних використовують криптографічні алгоритми:

- ДСТУ 7624:2014 (ECB, CBC, CFB, OFB, CTR);
- ГОСТ 28147-89 (ECB, CBC, CFB, OFB, CTR);
- AES (ECB, CBC, CFB, OFB, CTR);
- TDEA (ECB, CBC, CFB, OFB, CTR).

В основі лежить бібліотека «Шифр+» v2.1 (експертний висновок Держспецв'язку України 04/03/02-1674 от 16.05.2017).

Підтримувані СУБД Firebird v3.0 платформи наступні:

- 1) Windows: x86, x86-64;
- 2) Linux: x86, x86-64;
- 2) Android (experimental): ARM, ARM64.

8.4 Аналіз результатів проведення експеримента використання СУБД Firebird v3.0

Під час проведення експерименту з використання СУБД Firebird v3.0, було використано:

1) тип сервера:

- CPU: Intel Core i7-2600 CPU @ 3,40GHz;
- CPU Cores: 4 x 3,41 GHz (with HT – 8);
- HDD: 40GB;
- RAM: 4096 MB;
- OS: Centos 7 x86-64;
- Linux v.: 3.10.0-327.36.1.el7.x86_64;

2) СУБД: Firebird v3.0.2 x86-64;

3) клієнт:

- CPU: Intel Core i5-6400 CPU @ 2,70GHz;
- CPU Cores: 4 x 2,7 GHz;
- HDD: SATA III 2000GB;
- RAM: 8 GB;
- OS: Windows 10 (64-bit);

4) клієнт СУБД:

- IBExpert;
- v.2017.4.24.1;
- NET: Gigabit Ethernet;

5) БД: 7 таблиць, 28 індексів, 5 blob полів, 19 представлень, 5 тригерів, 1 генератор, 1 функція, 1 користувач;

6) розмір: 1008000 KB (~1 ГБ);

7) шифри:

- AES-NI-256;
- ГОСТ 28147-89 (ДКЕ №1);
- ДСТУ 7624:2014 (256/256).

Результатом проведення експерименту є первинне повне зашифрування БД розміром 1 Гб і подальше повне її розшифрування (табл. 1).

Таблиця 1

Отримані дані за проведеним експериментом

Algorithm-Mode	Производит., МБ/с		Время, с	
	Encrypt	Decrypt	Encrypt	Decrypt
AES-NI-256-OFB	109.14	110.131	9.01934	8.93823
AES-NI-256- CTR	101.696	101.882	9.67954	9.66193
ГОСТ 28147-OFB	46.4388	46.2943	21.1973	21.2634
ГОСТ 28147- CTR	44.7567	44.7679	21.9939	21.9884
ДСТУ 7624-256-OFB	66.1892	63.6726	14.8721	15.4599
ДСТУ 7624-256-CTR	66.0562	63.5132	14.9021	15.4987

Продуктивність СУБД за двома алгоритмами (табл. 2) залежить від етапів їхньої реалізації та полягає в такому.

Алгоритм 1	Алгоритм 2
1 – етап: SQL-скрипт створює порожню БД;	1 етап: створюється порожня БД;
2 етап: SQL-скрипт і Blob проводиться операція вставки великої кількості записів (понад 320000);	2 етап: шифрується одним з алгоритмів в одному з режимів;
3 етап: створюється резервна копія;	3 етап: проводиться операція вставки великої кількості записів;
4 етап: заміри часу операцій з даними на незашифрованій БД (вибірка, зміна, видалення)	4 етап: створюється резервна копія;
	5 етап: заміри часу операцій з даними в зашифрованій БД

Таблиця 2

Час виконання операцій (Время указано в формате mm:ss.fff)

Operation	Bulk Insert	Backup	Select	Update	Delete
None	25:23.610	00:17.000	00:04.688	00:08.672	00:08.578
AES-NI-256-OFB	25:43.922	00:26.000	00:04.719	00:10.516	00:08.734
AES-NI-256-CTR	26:09.344	00:25.000	00:04.703	00:10.843	00:08.890
ГОСТ28147-OFB	26:40.375	00:37.000	00:05.109	00:13.297	00:11.563
ГОСТ28147-CTR	25:49.547	00:36.000	00:05.31	00:13.766	00:12.000
ДСТУ7624-256-OFB	25:58.890	00:31.000	00:04.703	00:11.890	00:10.150
ДСТУ7624-256-CTR	25:54.969	00:33.000	00:04.782	00:12.000	00:10.172

Графічне представлення порівняння часу виконання основних запитів до бази даних з використанням СУБД Firebird v3.0 наведено на рис. 6.

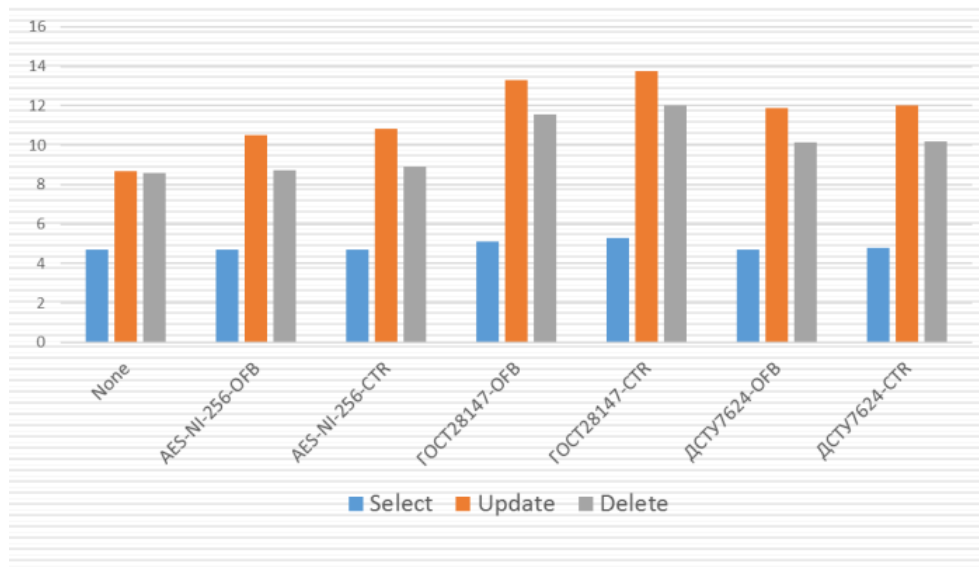


Рис. 6. Результат аналізу порівняння часу виконання основних запитів до бази даних з використанням СУБД Firebird v3.0

Графічне представлення порівняння часу завантаження великої кількості даних з використанням СУБД Firebird v3.0 наведено на рис. 7.

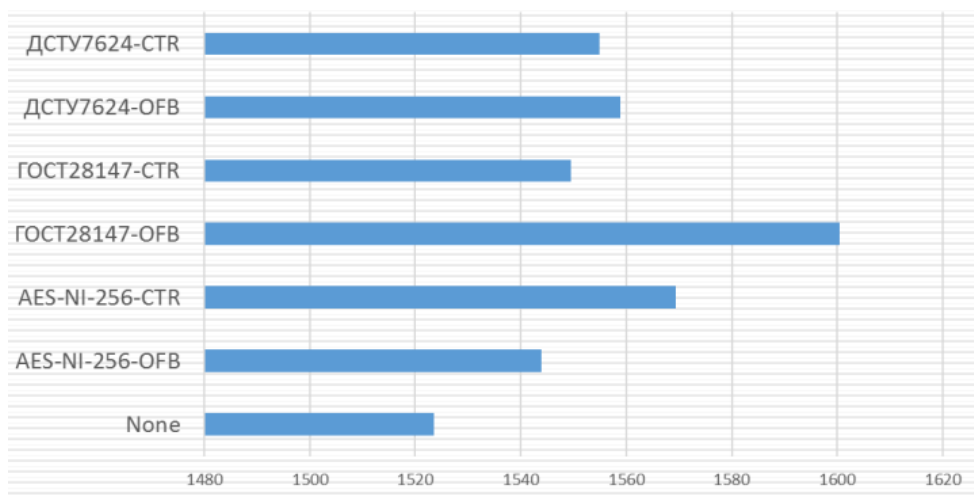


Рис. 7. Результат аналізу порівняння часу завантаження великої кількості даних з використанням СУБД Firebird v3.0

Графічне представлення порівняння часу створення резервної копії БД з використанням СУБД Firebird v3.0 наведено на рис. 8.

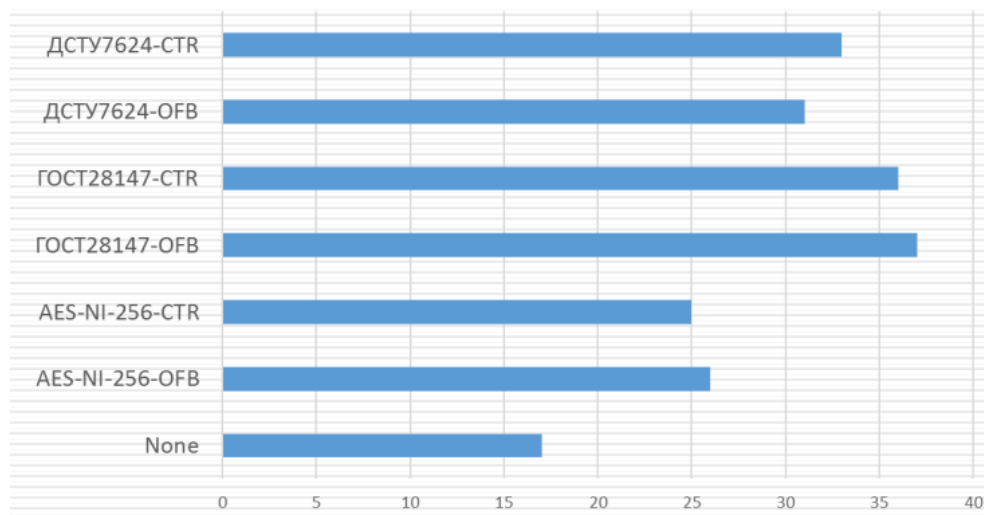


Рис. 8. Результат аналізу порівняння часу створення резервної копії БД з використанням СУБД Firebird v3.0

ЛАБОРАТОРНІ РОБОТИ

ЛАБОРАТОРНА РОБОТА № 1

СТВОРЕННЯ БАЗИ ДАНИХ У СЕРЕДОВИЩІ MS EXCEL

Мета:

Придбання навичок щодо створення таблиць бази даних та застосування основних функції з виконання запитів в додатку Excel.

Завдання:

1. Використовуючи засоби табличного процесора MS Excel, створити базу даних, що містить відомості про замовлення автомагазину за I квартал поточного року. Створену базу помістити на аркуш з ім'ям БД.

2. Доповнити базу даних «Замовлення автомагазину», що була створена в ході попереднього завдання, записами за березень місяць.

3. З використанням команди «Автофільтр» виконати пошук та вибірку записів бази даних «Замовлення автомагазину» спочатку за весь лютий місяць, а потім тільки за 2 лютого. Результати пошуку зберегти у файлі MS Word з ім'ям «Фільтр_ПБ» у вигляді скріншота вікна MS Excel.

4. З використанням команди «Автофільтру» здійснити пошук і вибірку п'яти найбільших «Цін (\$)». Результати пошуку зберегти у файлі MS Word з ім'ям «Фільтр_ПБ» у вигляді скріншота вікна MS Excel.

5. З використанням команди «Автофільтру», здійснити пошук замовлень, де «Ціна (\$)» вища за середню. Результати пошуку зберегти у файлі MS Word у вигляді скріншота вікна MS Excel.

6. З використанням команди «Автофільтру» знайти всі записи в базі даних, у яких «Ціна (\$)» більше 35 700, але менше 46 200. Результати пошуку зберегти у файлі MS Word у вигляді скріншота вікна MS Excel.

7. Здійснити одночасний пошук і вибірку всіх записів за чорним та червоним кольорами за допомогою команди «Автофільтру». Результати пошуку зберегти у файлі MS Word у вигляді скріншота вікна MS Excel.

8. За допомогою команди «Автофільтру» здійснити пошук і вибірку всіх записів у базі «Замовлення автомагазину», у яких значення поля «Оббивка» починається з букви В. Результати пошуку зберегти у файлі MS Word у вигляді скріншота вікна MS Excel.

9. Використовуючи команду «Розширений фільтр», зробити в базі даних «Замовлення автомагазину» три вибірки з даними: 1) за двадцять та двадцять восьме лютого; 2) колір «Антрацит» із «Ціною (\$)» більше або

рівною 39 200; 3) де значення поля «Ціна (\$)» більше 35 700 \$, але менше 46 200 \$.

Виконання завдання 1

Створення бази даних, що містить відомості про замовлення автомагазину за I квартал поточного року

1. Завантажити програму MS Excel. Створити нову книгу (якщо вона не створилася за замовчуванням). Змінити ім'я першого робочого аркуша на ім'я БД.

2. В комірку A1 ввести назву – «Замовлення за 1 квартал автомагазину "КЕНГО"». Відступити один рядок та ввести заголовний рядок з назвами полів згідно рис. 9.

	A	B	C	D	E	F	G	H	I
1	Замовлення за 1 квартал автомагазину "КЕНГО"								
2									
3	Модель кузова	Колір	Коробка передач	Оббивка	Ціна (\$)	№ замовлення	Прізвище, Ім'я, По батькові замовника	Телефон	Дата замовлення

Рис. 9. Ескіз таблиці «Замовлення за 1 квартал автомагазину "КЕНГО"»

Перед введенням записів встановити перевірку введення. Значення даних полів повинні задовольняти наступним умовам:

а. поле – Дата замовлення:

- тип даних – «Дата»;
- діапазон зміни – будь-яка дата, що відповідає I кварталу (1.01 – 31.03);
- повідомлення для введення – «введіть дату: I квартал (1.01 – 31.03)»;
- вид повідомлення про помилку – «Останов», текст – «Дата поза діапазоном»; введіть дату, що відповідає I кварталу.

б. поле – Колір:

- тип даних – «Список»;
- діапазон зміни – Червоний, Чорний, Антрацит;
- повідомлення для введення – «Виберіть зі списку колір».

с. поле – Коробка передач:

- тип даних – «Список»;
- діапазон зміни – Ручна, Автомат;

- повідомлення для введення – «Виберіть зі списку тип коробки передач».

d. поле – «Оббивка»:

- тип даних – «Список»;
- діапазон зміни – Шкіра, Велюр, Тканина;
- повідомлення для введення – «Виберіть зі списку тип оббивки».

e. поле – Ціна (\$):

- тип даних – «Действительное»;
- діапазон зміни – «больше 0»;
- повідомлення для введення – «Введіть дійсне число більшу нуля»;
- вид повідомлення про помилку – «Предупреждение», Сообщение: «Увага, тільки позитивне число».

3. Виконати форматування бази даних: об'єднати діапазон комірок A1:J1, встановити параметри шрифту для назви БД (розмір – 14, накреслення – напівжирний курсив); обрамувати заголовний рядок і встановити параметри шрифту (розмір – 12, накреслення – напівжирний).

4. Установити перевірку для поля «Дата замовлення». Для цього потрібно виділити діапазон комірок, куди будуть вводитися дати – A4:A9, і виконати команду « Проверка данных» (вкладка «Данные», група «Работа с данными»).

5. На вкладці «Параметры» вікна «Проверка вводимых значений», зі списку «Тип данных»: вибрати тип: «Дата» (рис. 10).

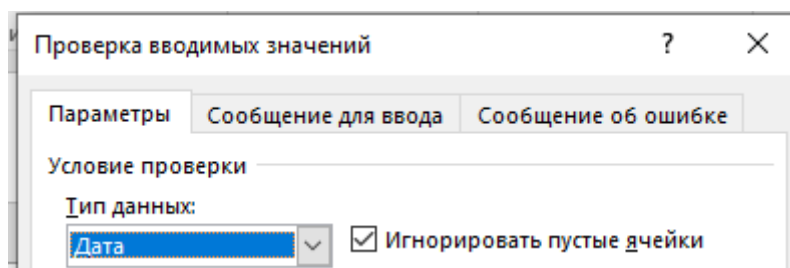


Рис. 10. Обрання значення поля «Тип данных:» вкладки «Параметры» для «Дата заповнення»

6. У списку «Значение» вибрати оператор порівняння для накладення обмеження на діапазон можливих значень даних: «между» (рис. 11).

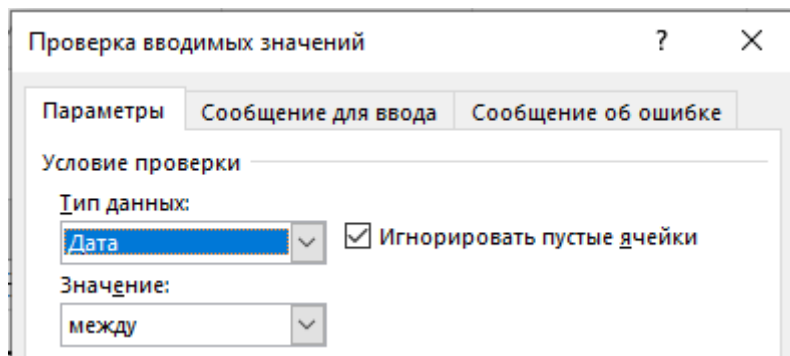


Рис. 11. Обрання значення поля «Значение:» вкладки «Параметры» для «Дата заповнення»

7. У поля для введення границ інтервалу можливих значень ввести в поле «Начальная дата:» – «1.01», у поле «Конечная дата:» – «31.03» (рис. 12).

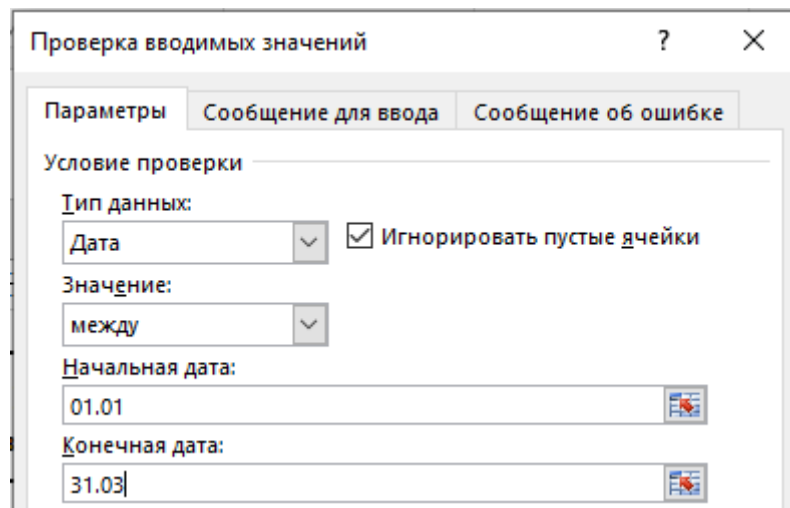


Рис. 12. Обрання значення полів «Начальная дата:» та «Конечная дата:» вкладки «Параметры» для «Дата заповнення»

8. Активізувати наступну вкладку «Сообщение для ввода». У поле «Заголовок» ввести текст: «Введіть дату:»; у поле «Сообщение» – I квартал року (1.01 – 31.03) (рис. 13).

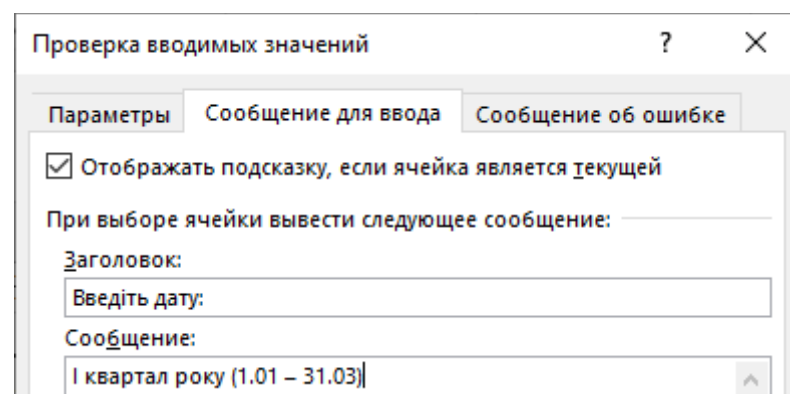


Рис. 13. Заповнення полів «Заголовок» та «Сообщение:» вкладки «Сообщение для ввода» для «Дата заповнення»

9. Активізувати вкладку «Сообщение об ошибке». У поле «Заголовок» ввести текст: «Дата поза діапазоном»; у поле «Сообщение» – ввести дату, що відповідає I кварталу. Зі списку «Вид» вибрати – «Останов». Клацнути на кнопці ОК (рис. 14).

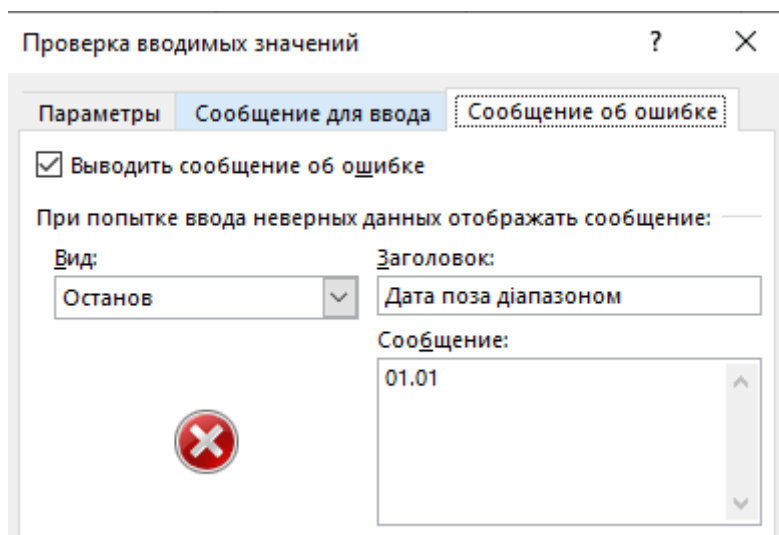


Рис. 14. Заповнення полів «Заголовок:» та «Вид:» вкладки «Сообщение об ошибке» для «Дата заповнення»

10. Щоб накласти обмеження на значення поля «Колір», спочатку на другому робочому аркуші в діапазоні комірок A1:A3 набрати список кольорів: Червоний, Чорний, Антрацит.

11. Присвоїти діапазону комірок A1:A3 довільне ім'я, наприклад, Колір. Для цього спочатку виділити діапазон A1:A3, а потім виконати команду «Присвоить Имя» (вкладка «Формулы», група «Определение имени»). У вікні «Создание имени» в поле «Имя» ввести: «Колір». Клацнути на кнопці ОК (рис. 15).

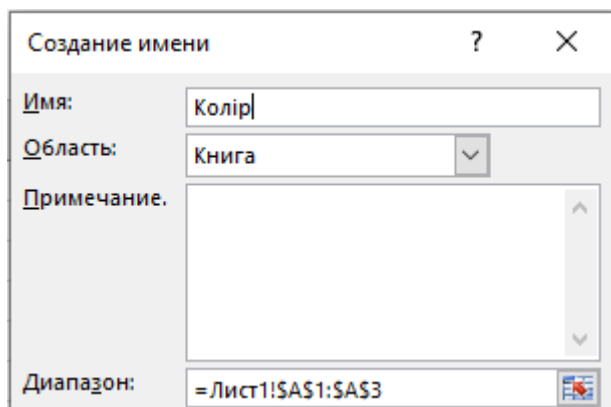


Рис. 15. Заповнення «Создание имени» для «Колір»

12. Повернутися на робочий аркуш з ім'ям БД, виділити діапазон комірок, куди будуть уводитися кольори, C4:C9, і виконати команду «Проверка данных» (вкладка «Данные», група «Работа с данными»).

13. На вкладці «Параметры» вікна «Проверка вводимых значений», в списку «Тип данных» вибрати тип: «Список» (рис. 16).

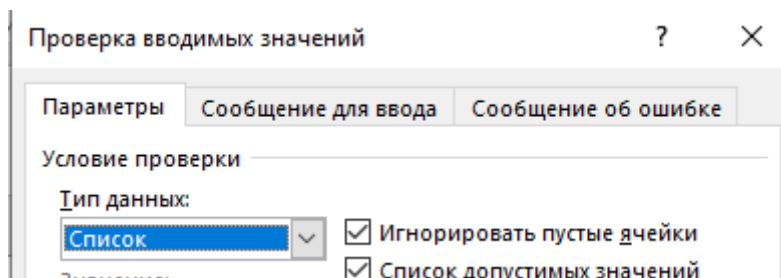


Рис. 16. Обрання значення поля «Тип данных:» вкладки «Параметры» для «Колір»

14. Встановити курсор у поле «Источник», перейти на вкладку «Формулы», у групі «Определенные имена» виконати команду «Использовать» в формуле та вибрати ім'я списку – «Колір» (рис. 17).

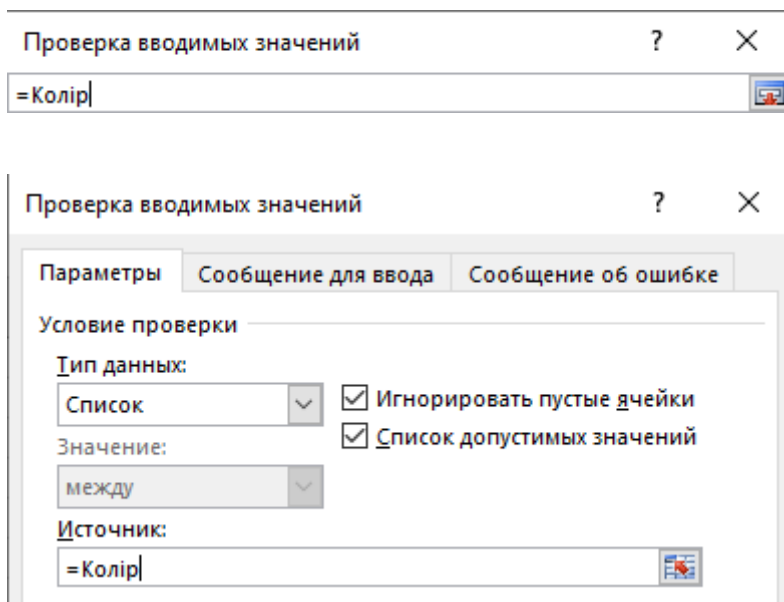


Рис. 17. Обрання типу для поля «Колір»

15. Активізувати вкладку «Сообщение для ввода» й ввести текст повідомлення: «Виберіть зі списку колір» (рис. 18).

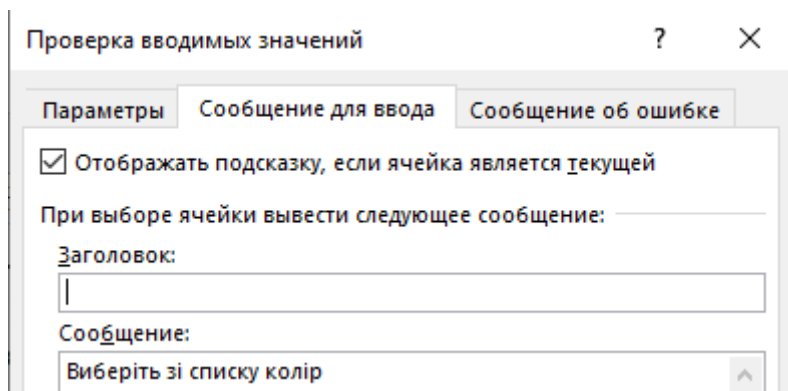


Рис. 18. Заповнення поля «Сообщение:» вкладки «Сообщение для ввода» для «Колір»

16. Самостійно встановити перевірку введення для полів бази даних «Коробка передач», «Оббивка» та «Ціна (\$)».

17. Ввести записи бази даних (рис. 19). Дані в поля «Колір», «Коробка передач» та «Оббивка» ввести за допомогою створеного списку, що розкривається.

17. Переконавшись в тому, що при активізації комірок бази даних, для яких встановлена перевірка введення, з'являться повідомлення для введення.

18. Ввести в поля «Дата замовлення», «Колір», «Коробка передач», «Оббивка» та «Ціна (\$)», що не належить інтервалу припустимих значень, і перевірити, чи з'явиться повідомлення про помилку.

19. Виконати необхідне форматування та налаштувати ширину стовбців та висоту рядків таблиці для коректного відображення даних.

20. Зберегти робочу книгу у власній папці на жорсткому диску під ім'ям «Замовлення автомагазину».

Виконання завдання 2

Доповнення бази даних записами

1. Відкрити робочу книгу «Замовлення автомагазину» та перейти на робочий аркуш з ім'ям БД.

2. Виділити рядок з останнім записом бази даних та виконати команду «Вставить – Вставить строки на лист» (вкладка «Главная», група «Ячейки»).

3. Діючи за аналогією, додати в базу даних 4 порожніх рядка.

4. Ввести нові записи (рис. 20).

Замовлення за 1 квартал автомагазину "КЕНГО"								
Модель кузова	Колір	Коробка передач	Оббивка	Ціна (\$)	№ замовлення	Прізвище, Ім'я, По батькові замовника	Телефон	Дата замовлення
12579 Classic liftback XL	Червоний	Ручна	Тканина	35700	123	Іванов Федор Степанович	8(066)2453456	03.янв
12580 Classic liftback GT	Чорний	Автомат	Тканина	39200	130	Петров Олег Іванович	8(067)7834512	06.янв
12651 Classic Compact XL	Антрацит	Ручна	Шкіра	41100	133	Сидоров Микола Сергійович	8(068)9004523	25.янв
12653 Classic Compact GT	Чорний	Ручна	Велюр	37900	135	Бендер Остап Ібрагімович	8(096)5743121	05.фев
12410 Classic Combi	Антрацит	Автомат	Тканина	46200	138	Іванов Сергій Сергійович	8(095)8674590	20.фев
12653 Classic Compact GT	Чорний	Ручна	Велюр	37900	140	Петров Юрій Андрійович	8(050)3482900	28.фев
12410 Classic Combi	Антрацит	Автомат	Тканина	46200	145	Сидоров Борис Борисович	8(066)4782390	05.мар
12580 Classic liftback GT	Чорний	Автомат	Тканина	39200	160	Дубов Дмитро Іванович	8(095)0045619	17.мар
12410 Classic Combi	Антрацит	Автомат	Тканина	46200	165	Сухов Олексій Олегович	8(067)3901211	20.мар
12653 Classic Compact GT	Чорний	Ручна	Велюр	37900	166	Сахаров Ігор Ігорович	8(096)6903478	25.мар

Рис. 19. Записи бази даних

Модель кузова	Колір	Коробка передач	Оббивка	Ціна (\$)	№ замовлення	Прізвище, Ім'я, По батькові замовника	Телефон	Дата замовлення
12410 Classic Combi	Антрацит	Автомат	Тканина	46200	145	Сидоров Борис Борисович	8(066)4782390	05.мар
12580 Classic liftback GT	Чорний	Автомат	Тканина	39200	160	Дубов Дмитро Іванович	8(095)0045619	17.мар
12410 Classic Combi	Антрацит	Автомат	Тканина	46200	165	Сухов Олексій Олегович	8(067)3901211	20.мар
12653 Classic Compact GT	Чорний	Ручна	Велюр	37900	166	Сахаров Ігор Ігорович	8(096)6903478	25.мар
12410 Classic Combi	Антрацит	Автомат	Тканина	46200	138	Іванов Сергій Сергійович	8(095)8674590	31.мар

Рис. 20. Записи за березень

Виконання завдання 3

Виконання пошуку та виборки записів бази даних за вказані періоди

1. Встановити режим Автофільтру (помістити рамку виділення у будь-яку комірку БД та виконати команду «Данные» – «Фильтр») (рис. 21).

Модель кузова	Колір	Коробка передач	Оббивка	Ціна (\$)	№ замовлення	Прізвище, Ім'я, По батькові замовника	Телефон	Дата замовлення
---------------	-------	-----------------	---------	-----------	--------------	---------------------------------------	---------	-----------------

Рис. 21. Встановлення автофільтру

2. Відкрити список кнопкою автофільтру для поля «Дата замовлення» та встановити галочку перед відповідним місяцем (Лютий) інші позначки зняти. Натиснути кнопку ОК.

3. У результаті на екрані залишаються тільки ті записи, дата в яких відповідає встановленому місяцю (рис. 22).

	A	B	C	D	E	F	G	H	I
1	Замовлення за 1 квартал автомагазину "КЕНГО"								
2									
3	Модель кузова	Колір	Коробка передач	Оббивка	Ціна (\$)	№ замовлення	Прізвище, Ім'я, По батькові замовника	Телефон	Дата замовлення
7	12653 Classic Compact GT	Чорний	Ручна	Велюр	37900	135	Бендер Остап Ібрагімович	8(096)5743121	05.фев
8	12410 Classic Combi	Антрацит	Автомат	Тканина	46200	138	Іванов Сергій Сергійович	8(095)8674590	20.фев
9	12653 Classic Compact GT	Чорний	Ручна	Велюр	37900	140	Петров Юрій Андрійович	8(050)3482900	28.фев

Рис. 22. Результат автофільтру поля «Дата замовлення» за лютим

4. Зробити скріншот вікна MS Excel (клавіша Alt+PrtSc) та вставити його в документ MS Word.

5. Відновити базу за допомогою опції «Удалить фильтр из столбца «Дата замовлення», або командою «Данные – Очистить» (режим автофільтру при цьому залишається).

6. Знову відкрити список, що розкривається для поля «Дата замовлення» (рис. 15) та розкрити список «Лютий». Встановити галочку перед відповідною датою (05 лютого), інші позначки зняти. Натиснути кнопку ОК (рис. 23).

Модель кузова	Колір	Коробка передач	Оббивка	Ціна (\$)	№ замовлення	Прізвище, Ім'я, По батькові замовника	Телефон	Дата замовлення
12653 Classic Compact GT	Чорний	Ручна	Велюр	37900	135	Бендер Остап Ібрагімович	8(096)5743121	05.фев

Рис. 23. Результат автофільтрування записів за датою 05 лютого

7. У результаті на екрані залишаються тільки ті записи, дата в яких відповідає 05 лютому. Зробити скріншот вікна MS Excel та вставити його в документ MS Word з ім'ям «Фільтр_ПБ».

8. Відновити базу за допомогою опції «Удалить фильтр из столбца «Дата замовлення», або командою «Данные – Очистить».

Виконання завдання 4

Здійснення пошуку і виборки певної кількості значень за критерієм

Наложение условия по списку ? X

Показать

наибольших 5 элементов списка

OK Отмена

Модель кузова	Колір	Коробка передач	Оббивка	Ціна (\$)	№ замовлення	Прізвище, Ім'я, По батькові замовника	Телефон	Дата замовлення
12580 Classic liftback GT	Чорний	Автомат	Тканина	39200	130	Петров Олег Іванович	8(067)7834512	06.янв
12651 Classic Compact XL	Антрацит	Ручна	Шкіра	41100	133	Сидоров Микола Сергійович	8(068)9004523	25.янв
12653 Classic Compact GT	Чорний	Ручна	Велюр	37900	135	Бендер Остап Ібрагімович	8(096)5743121	05.фев
12410 Classic Combi	Антрацит	Автомат	Тканина	46200	138	Іванов Сергій Сергійович	8(095)8674590	20.фев
12653 Classic Compact GT	Чорний	Ручна	Велюр	37900	140	Петров Юрій Андрійович	8(050)3482900	28.фев
12410 Classic Combi	Антрацит	Автомат	Тканина	46200	145	Сидоров Борис Борисович	8(066)4782390	05.мар
12580 Classic liftback GT	Чорний	Автомат	Тканина	39200	160	Дубов Дмитро Іванович	8(095)0045619	17.мар
12410 Classic Combi	Антрацит	Автомат	Тканина	46200	165	Сухоп Олексій Олександрович	8(067)3901211	20.мар
12653 Classic Compact GT	Чорний	Ручна	Велюр	37900	166	Сахаров Ігор Ігорович	8(096)6903478	25.мар
12410 Classic Combi	Антрацит	Автомат	Тканина	46200	138	Іванов Сергій Сергійович	8(095)8674590	31.мар

Рис. 24. Результат числового автофільтрування за встановленим критерієм щодо п'яти записів з найбільшими значеннями ціни

1. Переконалися в тому що в базі даних відображені все записи. Якщо ні, відновити усі записи командою «Данные – Очистить».

2. Відкрити список, що розкривається, для поля «Ціна (\$)» та вибрати команду «Числовые фильтры – Первые 10 ...».

3. Використовуючи діалогове вікно «Наложение условия по списку» (рис 3.3), встановити параметри для вибору 5 найбільших елементів списку та натиснути кнопку ОК (рис. 24).

4. Переконалися, що на екрані залишились не менше п'яти записів, де значення ціни максимальне.

5. Зробити скріншот вікна MS Excel (клавіша Alt+PrtSc) та вставити його в ваш документ MS Word.

6. Відновити базу за допомогою опції «Удалить фильтр из столбца «Дата замовлення» або командою «Данные – Очистить».

Виконання завдання 5

Здійснення пошуку замовлень, де «Ціна (\$)» вища за середню

1. Переконалися в тому що в базі даних відображені все записи. Якщо ні, відновити усі записи командою «Данные – Очистить».

2. Відкрити список, що розкривається для поля «Ціна (\$)» та вибрати команду «Числовые фильтры – Выше среднего».

3. Переконалися, що на екрані залишились записи, які задовольняють умовам фільтрації (рис. 25).

Модель кузова	Колір	Коробка передач	Оббивка	Ціна (\$)	№ замовлення	Прізвище, Ім'я, По батькові замовник	Телефон	Дата замовлення
12410 Classic Combi	Антрацит	Автомат	Тканина	46200	138	Іванов Сергій Сергійович	8(095)86745 90	20.фев
12410 Classic Combi	Антрацит	Автомат	Тканина	46200	145	Сидоров Борис Борисович	8(066)47823 90	05.мар
12410 Classic Combi	Антрацит	Автомат	Тканина	46200	165	Сузов Олексій Олегович	8(067)39012 11	20.мар
12410 Classic Combi	Антрацит	Автомат	Тканина	46200	138	Іванов Сергій Сергійович	8(095)86745 90	31.мар

Рис. 25. Результати числового автофільтрування за встановленим критерієм значення ціни вище середнього

4. Зробити скріншот вікна MS Excel (клавіша Alt+PrtSc) та вставити його в документ MS Word з ім'ям «Фільтр_ПІБ».

5. Відновити базу за допомогою опції «Удалить фильтр из столбца «Дата замовлення» або командою «Данные – Очистить».

Виконання завдання 6

Знаходження записів в базі даних з обмеженнями

1. Переконалися в тому, що в базі даних відображені всі записи. Якщо ні, відновити усі записи командою «Данные – Очистить».

2. В списку поля «Ціна (\$)» активізувати команду «Числовые фильтры – Настраиваемый фильтр...», на екрані з'явиться вікно «Пользовательский автофильтр» (рис. 26).

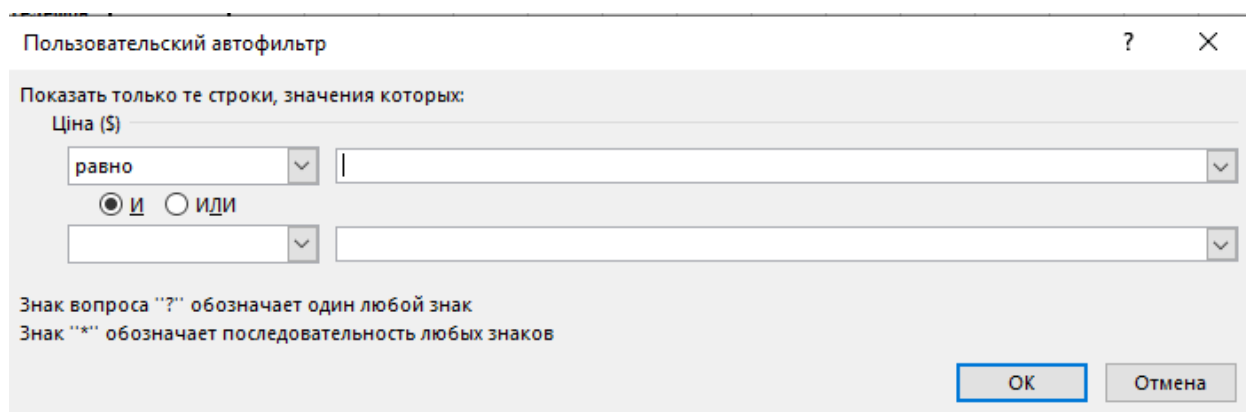


Рис. 26. Вікно «Пользовательский автофильтр»

3. В першому рядку зі списку, що розкривається вибрати – «больше», а у перший комбінований рядок введення критеріїв ввести – «35 700».

4. У другому рядку зі списку, що розкривається вибрати – «меньше», а у другий комбінований рядок введення критеріїв ввести – «46 200».

5. Критерії зв'язати функцією «И» за допомогою відповідного перемикача (рис. 27).



Рис. 27. Заповнення вікна «Пользовательский автофильтр»

6. Операцію завершити клацанням на кнопці ОК (рис. 28).

Модель кузова	Колір	Коробка передач	Оббивка	Ціна (\$)	№ замовлення	Прізвище, Ім'я, По батькові замовника	Телефон	Дата замовлення
12580 Classic liftback GT	Чорний	Автомат	Тканина	39200	130	Петров Олег Іванович	8(067)7834512	06.янв
12651 Classic Compact XL	Антрацит	Ручна	Шкіра	41100	133	Сидоров Микола Сергійович	8(068)9004523	25.янв
12653 Classic Compact GT	Чорний	Ручна	Велюр	37900	135	Бендер Остап Ібрагімович	8(096)5743121	05.фев
12653 Classic Compact GT	Чорний	Ручна	Велюр	37900	140	Петров Юрій Андрійович	8(050)3482900	28.фев
12580 Classic liftback GT	Чорний	Автомат	Тканина	39200	160	Дубов Дмитро Іванович	8(095)0045619	17.мар
12653 Classic Compact GT	Чорний	Ручна	Велюр	37900	166	Сахаров Ігор Ігорович	8(096)6903478	25.мар

Рис. 28. Результат виконання запиту

7. Перевірити результат, зробити скріншот вікна MS Excel та вставити його в документ MS Word з ім'ям «Фільтр_ПБ». Відновити базу.

Виконання завдання 7

Здійснення одночасного пошуку і виборки всіх записів за певними критеріями

1. Переконалися в тому, що в базі даних відображені всі записи. Якщо ні, відновити усі записи командою «Данные – Очистить».

2. Активізувати команду «Текстовые фильтры – Настраиваемый фильтр...», зі списку, що розкривається, поля «Колір», на екрані з'явиться вікно «Пользовательский автофильтр» (рис. 29).

Рис. 29. Вікно «Пользовательский автофильтр»

3. У вікні «Пользовательский автофильтр» із першого й другого списків вибрати – «равно».

4. У першому комбінованому рядку введення критеріїв вибрати – «Червоний», у другому – «Антрацит».

5. Критерії зв'язати функцією «ИЛИ» за допомогою відповідного перемикача (рис. 30).

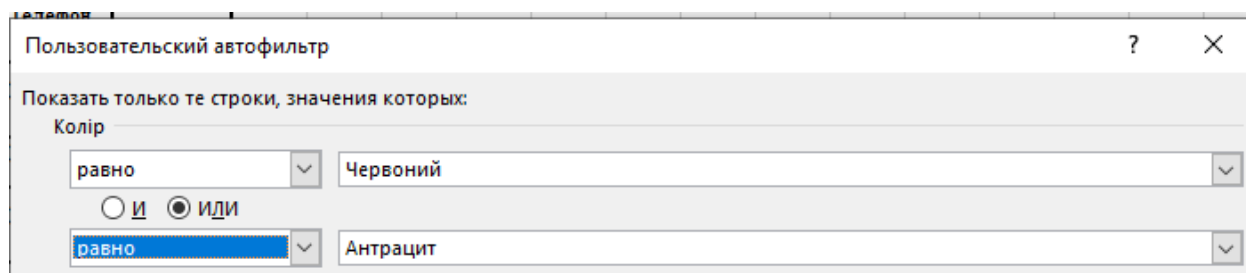


Рис. 30. Заповнення вікна «Пользовательский автофильтр»

6. Операцію завершити клацанням на кнопці ОК (рис. 31).

Модель кузова	Колір	Коробка передач	Оббивка	Ціна (\$)	№ замовлення	Прізвище, Ім'я, По батькові замовник	Телефон	Дата замовлення
12579 Classic liftback XL	Червоний	Ручна	Тканина	35700	123	Іванов Федор Степанович	8(066)24534 56	03.янв
12651 Classic Compact XL	Антрацит	Ручна	Шкіра	41100	133	Сидоров Микола Сергійович	8(068)90045 23	25.янв
12410 Classic Combi	Антрацит	Автомат	Тканина	46200	138	Іванов Сергій Сергійович	8(095)86745 90	20.фев
12410 Classic Combi	Антрацит	Автомат	Тканина	46200	145	Сидоров Борис Борисович	8(066)47823 90	05.мар
12410 Classic Combi	Антрацит	Автомат	Тканина	46200	165	Суков Олексій Олегович	8(067)39012 11	20.мар
12410 Classic Combi	Антрацит	Автомат	Тканина	46200	138	Іванов Сергій Сергійович	8(095)86745 90	31.мар

Рис. 31. Результат виконання запиту

7. Перевірити результат, зробити скріншот вікна MS Excel та вставити його в ваш документ MS Word. Відновити базу.

Виконання завдання 8

Здійснення пошуку і виборки всіх записів у базі, в яких значення поля починається з встановленої букви

1. Переконалися в тому що в базі даних відображені все записи. Якщо ні, відновити усі записи командою «Данные – Очистить».

2. Активізувати команду «Текстовые фильтры – начинается с...», зі списку, що розкривається, поля «Оббивка», на екрані з'явиться вікно «Пользовательский автофильтр» (рис. 32).

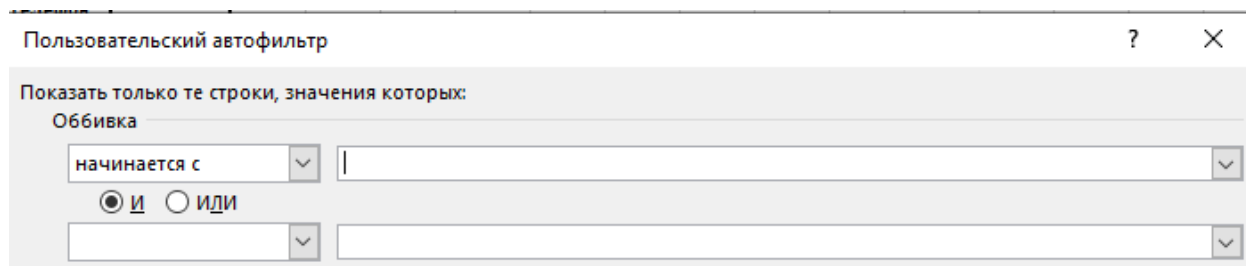


Рис. 32. Вікно «Пользовательский автофильтр»

3. Переконалися, що у вікні «Пользовательский автофильтр» у першому списку встановлено операцію «начинается с» та ввести в поле введення критеріїв букву – В (рис. 33).



Рис. 33. Заповнення вікна «Пользовательский автофильтр»

4. Операцію завершити клацанням на кнопці ОК (рис. 34).

Модель кузова	Колір	Коробка передач	Оббивка	Ціна (\$)	№ замовлення	Прізвище, Ім'я, По батькові замовника	Телефон	Дата замовлення
12653 Classic Compact GT	Чорний	Ручна	Велюр	37900	135	Бендер Остап Ібрагімович	8(096)57431 21	05.фев
12653 Classic Compact GT	Чорний	Ручна	Велюр	37900	140	Петров Юрій Андрійович	8(050)34829 00	28.фев
12653 Classic Compact GT	Чорний	Ручна	Велюр	37900	166	Сахаров Ігор Ігорович	8(096)69034 78	25.мар

Рис. 34. Результат виконання запиту

5. Перевірити результат, зробити скріншот вікна MS Excel та вставити його в ваш документ MS Word. Відновити базу.

Виконання завдання 9

За допомогою розширеного фільтру, зробити в базі даних три вибірки з даними

1. Для здійснення пошуку необхідно спочатку створити 3 блоки критеріїв на Робочому аркуші з базою даних. Блок критеріїв створити праворуч від бази даних «Замовлення автомагазину» К і L (діапазони K4:K6, K8:L9, K11:L12). Назви полів за якими буде здійснюватися розташування даних в результаті пошуку в базі даних «Замовлення автомагазину». В комірки за назвою поля ввести значення критерію для пошуку (рис. 35).

К	L
Дата замовлення	
20.фев	
28.фев	
Колір	Ціна (\$)
Антрацит	>=39200
Сума непрямого збитку	Сума непрямого збитку
>37900	<46200

Рис. 35. Критерії пошуку

2. Рамку виділення встановити в будь-яку комірку бази даних і виконати команду «Данные – Дополнительно». На екрані з'явиться однойменне діалогове вікно (рис. 36).

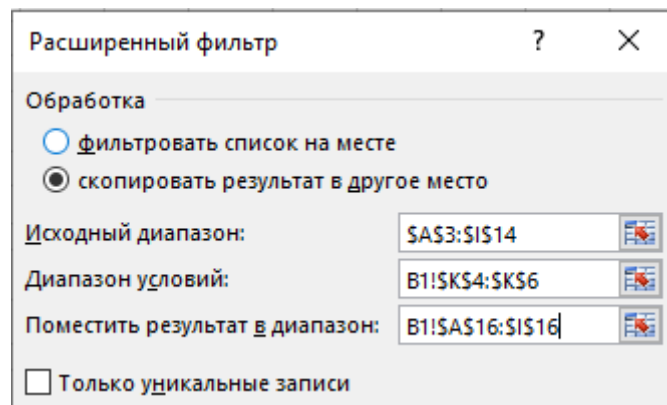


Рис. 36. Результат виконання команди «Данные – Дополнительно»

3. У діалоговому вікні встановити:

- перемикач «скопировать результат в другое место»;
- «Исходный диапазон:» розміщення бази даних встановити – \$A\$3:\$I\$14 (тобто всю БД з рядком заголовку включно);

- у поле «Діапазон умовий»: ввести діапазон розміщення першого критерію – \$K\$4:\$K\$6 (тобто перший блок критеріїв пошуку з назвою поля включно);

- у поле «Поместить результат в диапазон:» ввести посилання на першу комірку діапазону виводу результату пошуку й вибірки – \$A\$16:\$I\$16.

4. Клацнути на кнопці ОК.

Знизу бази даних починаючи з комірки \$A\$16 з'явиться результат вибірки (рис. 37).

	Модель кузова	Колір	Коробка передач	Оббивка	Ціна (\$)	№ замовлення	Прізвище, Ім'я, По батькові замовника	Телефон	Дата замовлення
16									
17	12410 Classic Combi	Антрацит	Автомат	Тканина	46200	138	Іванов Сергій Сергійович	8(095)8674590	20.фев
18	12653 Classic Compact GT	Чорний	Ручна	Велюр	37900	140	Петров Юрій Андрійович	8(050)3482900	28.фев

Рис. 37. Результат вибірки даних замовлень автомагазину за встановлені дати

5. Пошук записів з даними, що задовольняють другому критерію у базі даних «Замовлення автомагазину» виконати аналогічно. У вікні «Расширенный фильтр» ввести нові посилання на другий блок критеріїв й діапазону виводу:

- «Діапазон умовий:» – \$K\$8:\$L\$9;
- «Поместить результат в диапазон:» – \$A\$21:\$I\$21 (рис. 38).

Рис. 38. Налаштування розширеного фільтра для виконання другого запиту

Результат запиту наведено на рис. 39.

Модель кузова	Колір	Коробка передач	Оббивка	Ціна (\$)	№ замовлення	Прізвище, Ім'я, По батькові замовника	Телефон	Дата замовлення
12651 Classic Compact XL	Антрацит	Ручна	Шкіра	41100	133	Сидоров Микола Сергійович	8(068)9004523	25.янв
12410 Classic Combi	Антрацит	Автомат	Тканина	46200	138	Іванов Сергій Сергійович	8(095)8674590	20.фев
12410 Classic Combi	Антрацит	Автомат	Тканина	46200	145	Сидоров Борис Борисович	8(066)4782390	05.мар
12410 Classic Combi	Антрацит	Автомат	Тканина	46200	165	Сухов Олексій Олегович	8(067)3901211	20.мар
12410 Classic Combi	Антрацит	Автомат	Тканина	46200	138	Іванов Сергій Сергійович	8(095)8674590	31.мар

Рис. 39. Результат запиту за другим завданням

6. Для пошуку запису по третьому критерію у вікні «Расширенный фильтр» ввести наступні посилання на третій блок критеріїв й діапазону виводу:

- «Диапазон условий»: – \$K\$11:\$L\$12;
- «Поместить результат в диапазон:» – \$A\$29:\$I\$29 (рис. 40).

Расширенный фильтр

Обработка

☐ фильтровать список на месте

☒ скопировать результат в другое место

Исходный диапазон: \$A\$3:\$I\$14

Диапазон условий: B1:\$K\$11:\$L\$12

Поместить результат в диапазон: B1:\$A\$29

☐ Только уникальные записи

OK Отмена

Рис. 40. Налаштування розширеного фільтру для виконання третього запиту

7. Перевірити результат та зробити висновки (рис. 41).

Модель кузова	Колір	Коробка передач	Оббивка	Ціна (\$)	№ замовлення	Прізвище, Ім'я, По батькові замовника	Телефон	Дата замовлення
12580 Classic liftback GT	Чорний	Автомат	Тканина	39200	130	Петров Олег Іванович	8(067)7834512	06.янв
12651 Classic Compact XL	Антрацит	Ручна	Шкіра	41100	133	Сидоров Микола Сергійович	8(068)9004523	25.янв
12653 Classic Compact GT	Чорний	Ручна	Велюр	37900	135	Бендер Остап Ібрагімович	8(096)5743121	05.фев
12653 Classic Compact GT	Чорний	Ручна	Велюр	37900	140	Петров Юрій Андрійович	8(050)3482900	28.фев
12580 Classic liftback GT	Чорний	Автомат	Тканина	39200	160	Дубов Дмитро Іванович	8(095)0045619	17.мар
12653 Classic Compact GT	Чорний	Ручна	Велюр	37900	166	Сахаров Ігор Ігорович	8(096)6903478	25.мар

Рис. 41. Результат запиту за третім завданням

Контрольні запитання та завдання

1. З чого складається критерій пошуку? Що є результатом перевірки критерію?
2. Які шаблони використовуються при пошуку текстових фрагментів?
3. Які недоліки існують при використанні вікна «Форма»?
4. Для чого призначена команда «Автофільтр» та «Расширенный фильтр»?
5. Які логічні функції використовуються при об'єднанні декількох критеріїв пошуку?
6. Які переваги та недоліки кожного виду фільтрів?

ЛАБОРАТОРНА РОБОТА № 2

ВСТАНОВЛЕННЯ СУБД MYSQL ТА СУБД POSTGRESQL

Мета:

Придбання навичок щодо встановлення та розгортання запропонованої редакції системи управління базами даних (СУБД) MySQL та вільної об'єктно-реляційної системи управління базами даних (СУБД) PostgreSQL.

Завдання:

1. За наведеними посиланнями провести завантаження, встановлення, налаштування та розгортання СУБД MySQL на Windows 10.
2. За наведеними посиланнями провести завантаження, встановлення, налаштування та розгортання СУБД PostgreSQL (SQL Shell) на Windows 10.
3. За наведеними посиланнями провести завантаження, встановлення, налаштування та розгортання СУБД PostgreSQL (pgAdmin) на Windows 10.

Виконання завдання 1

Завантаження, встановлення, налаштування та розгортання СУБД MySQL на Windows 10

MySQL 8 випускається в декількох редакціях.

MySQL Standard Edition

MySQL Standard Edition – це редакція зі стандартним функціоналом, вона дає змогу реалізувати високопродуктивні, надійні та масштабовані додатки.

Вона містить повний функціонал і набір додаткових інструментів, які призначені для досягнення найвищої масштабованості, безпеки та надійності.

MySQL Cluster Carrier Grade Edition

MySQL Cluster (CGE) – це розподілена база даних, що поєднує лінійну масштабованість і високу доступність. Вона призначена для критично важливих додатків.

MySQL Community

MySQL Community Edition – це безкоштовна редакція, яку можна абсолютно вільно завантажити з офіційного сайту MySQL. Редакція підтримує весь необхідний функціонал для реалізації повноцінних Інтернет-ресурсів.

Покрокове встановлення MySQL 8 на Windows 10

Крім сервера MySQL встановимо і MySQL Workbench – це інструмент, за допомогою якого здійснюється розробка баз даних і адміністрування MySQL Server.

Примітка. Для встановлення MySQL 8 на Windows 10 потрібне середовище Microsoft .NET Framework 4.5.2 або новіша версія.

Крок 1 – Завантаження програми MySQL 8 для Windows 10

Щоб перейти до встановлення MySQL 8, спочатку необхідно завантажити дистрибутив цієї програми.

Завантажити MySQL 8 в редакції Community можна, як було вже зазначено, абсолютно вільно з офіційного сайту [https://dev.mysql.com/downloads/installer].

Після переходу на сторінку можна вибрати спосіб завантаження дистрибутива, нам пропонують два способи:

- завантажити Web-інсталятор – невелика за розміром програма, яка завантажує всі необхідні компоненти для встановлення MySQL. Можна використовувати для встановлення MySQL на комп'ютерах, де є Інтернет;
- завантажити повний інсталятор – дистрибутив програми MySQL, який містить усі необхідні компоненти. У цьому випадку можна використовувати цей дистрибутив для встановлення MySQL на комп'ютері як з доступом, так і без доступу до Інтернету. Наприклад, можна завантажити цей файл на комп'ютері, де є Інтернет, а використовувати його для встановлення MySQL на іншому комп'ютері, який без Інтернету.

Для приклада, завантажити повний дистрибутив програми MySQL, для цього натискаємо «Download» навпроти відповідного пункту (рис. 42).

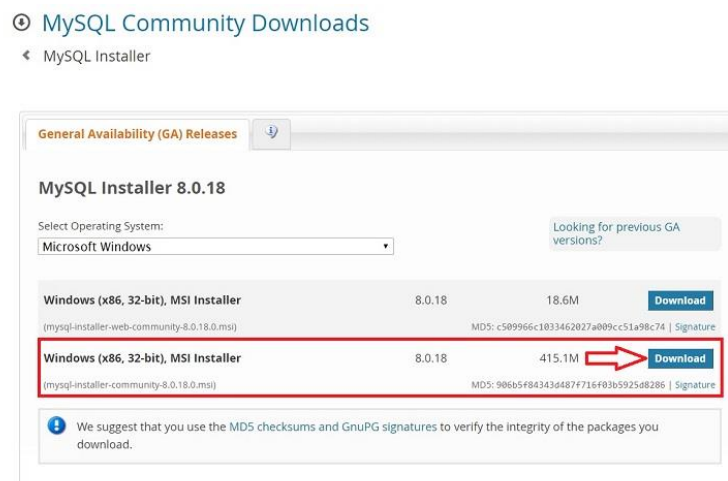


Рис. 42. Завантаження повного дистрибутиву програми MySQL

Потім пропонується авторизуватися, використовуючи облікові дані Oracle, якщо вони є, або зареєструватися, тим самим створити обліковий запис Oracle. Однак якщо цього не зробити, то в нижній частині є посилання «No thanks, just start my download», натиснувши на яке одразу почнеться завантаження MySQL. Натиснути на це посилання і зачекати закінчення завантаження (рис. 43).

MySQL Community Downloads

Login Now or Sign Up for a free account.

An Oracle Web Account provides you with the following advantages:

- Fast access to MySQL software downloads
- Download technical White Papers and Presentations
- Post messages in the MySQL Discussion Forums
- Report and track bugs in the MySQL bug system

Login »
using my Oracle Web account

Sign Up »
for an Oracle Web account

MySQL.com is using Oracle SSO for authentication. If you already have an Oracle Web account, click the Login link. Otherwise, you can signup for a free account by clicking the Sign Up link and following the instructions.

No thanks, just start my download.



Рис. 43. Завантаження MySQL

У результаті має завантажитися файл **mysql-installer-community-8.0.18.0.msi** розміром приблизно 415 мегабайт.

Крок 2 – Запустити встановлення та вибір типу встановлення MySQL

Далі запустити скачаний файл, у результаті запуститься програма встановлення MySQL. Спочатку вибрати тип встановлення. Пропонується кілька варіантів:

- **Developer** – це варіант за замовчуванням, він передбачає встановлення всього того, що потрібно розробнику, це і MySQL Server, і MySQL Workbench та інші інструменти для роботи з MySQL;

- **Server Only** – цей варіант передбачає встановлення тільки сервера MySQL;

- Client Only – цей варіант передбачає встановлення тільки клієнтської частини для роботи з MySQL Server, тобто серверна частина встановлюватися не буде;

- Full – встановлення всіх включених в дистрибутив компонентів;

- Custom – вибіркове встановлення, у цьому разі можна встановити тільки те, що потрібно.

Варіант за замовчуванням, тобто Developer, підійде для більшості випадків, особливо під час встановлення MySQL на домашньому комп'ютері для якихось своїх цілей (вивчення MySQL, вивчення SQL тощо). Після цього натиснути «Next» (рис. 44).

Примітка. Інтерфейс програми встановлення, як і інтерфейс програми MySQL Workbench англійською мовою.

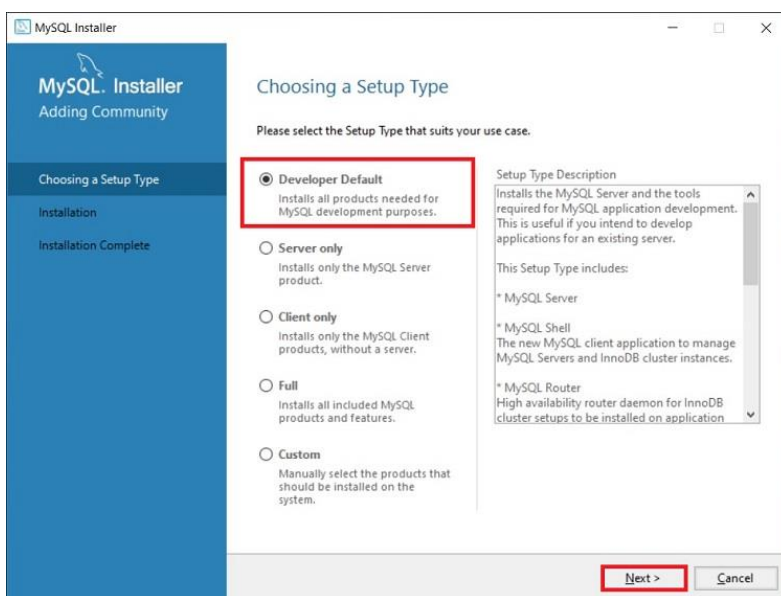


Рис. 44. Обрання Developer під час встановлення MySQL

Крок 3 – Перевірка та встановлення додаткових компонентів

Потім програма встановлення перевірить систему на наявність компонентів, які потрібні для роботи деяких розширень MySQL. У даному випадку програма встановлення видала два попередження для розширень MySQL For Excel і MySQL For Visual Studio. Якщо планується користуватися цими компонентами, то потрібно усунути ці зауваження. Для цього необхідно встановити ці компоненти (тобто виділити їх і натиснути «Execute», деякі можуть встановитися й автоматично).

Однак, якщо не використовуються розширення MySQL For Excel та MySQL For Visual Studio, то можна одразу натиснути «Next» (рис. 45).

Примітки. Встановлення Visual Studio 2019 Community на Windows 10.

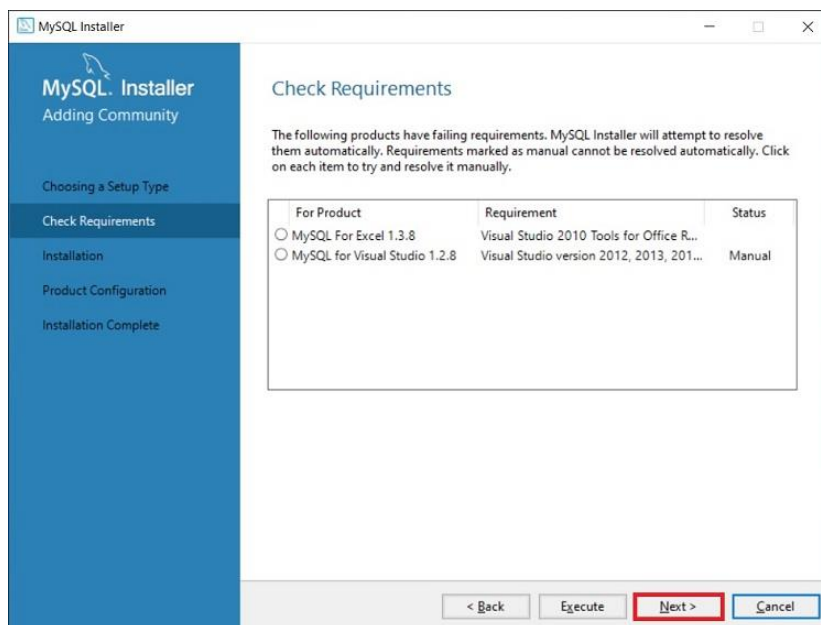


Рис. 45. Встановлення попереджень для розширень MySQL For Excel і MySQL For Visual Studio

Програма встановлення видасть попередження, пов'язане з відсутністю деяких компонентів, тиснемо «Yes» (рис. 46).

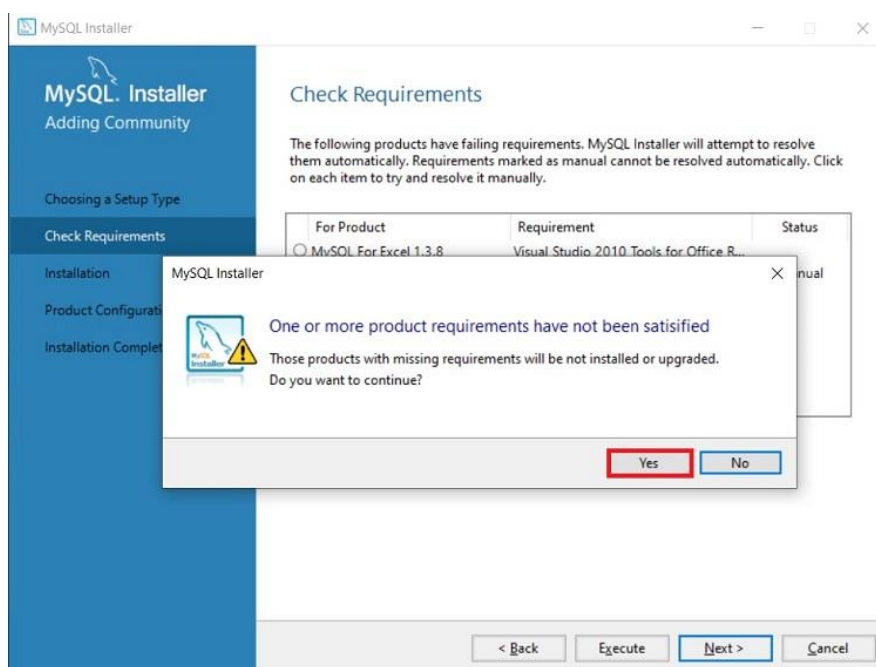


Рис. 46. Встановлення попередження, пов'язаного із відсутністю деяких компонентів

Крок 4 – Встановлення компонентів MySQL

Усе готово для встановлення MySQL, на цьому кроці програма встановлення видасть усі компоненти MySQL, які готові до встановлення, натискаємо «Execute» (рис. 47).

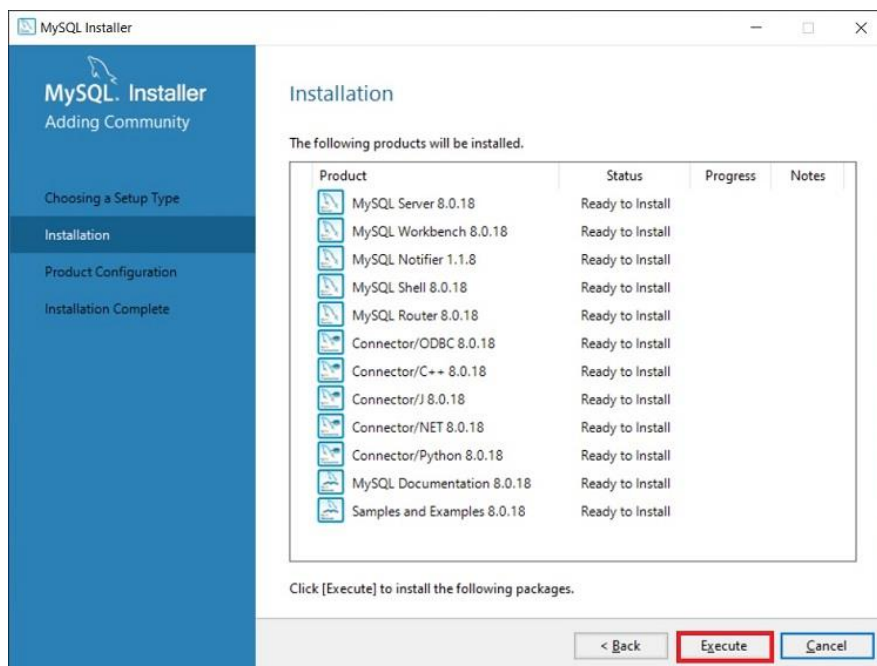


Рис. 47. Компоненти MySQL, які готові до встановлення

У результаті почнеться послідовне встановлення компонентів MySQL (рис. 48).

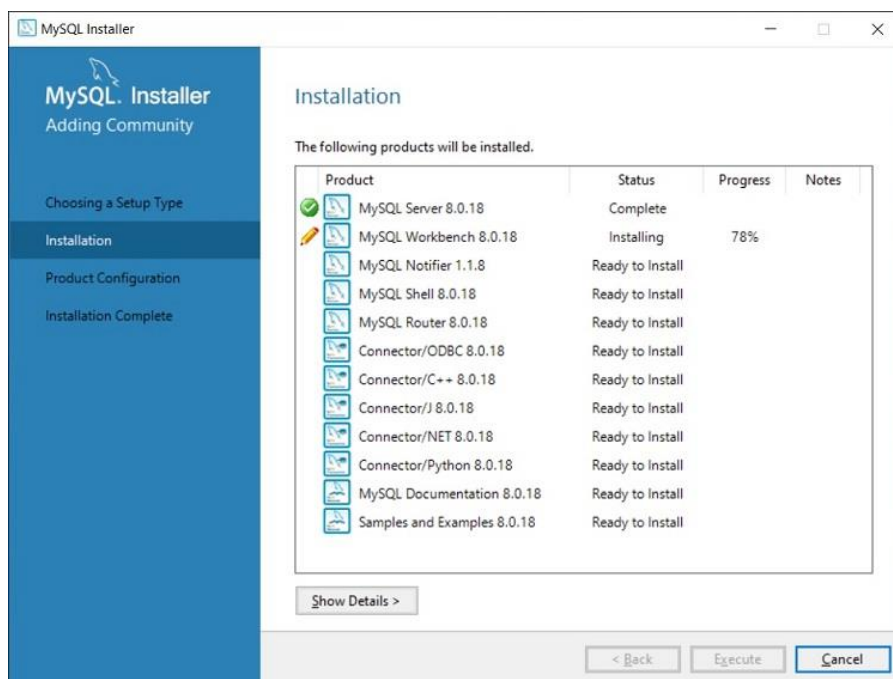


Рис. 48. Послідовне встановлення компонентів MySQL

Крок 5 – Завершення встановлення компонентів MySQL

Коли навпроти кожного пункту з'явиться зелена галочка, встановлення буде завершено. Натискаємо «Next» (рис. 49).

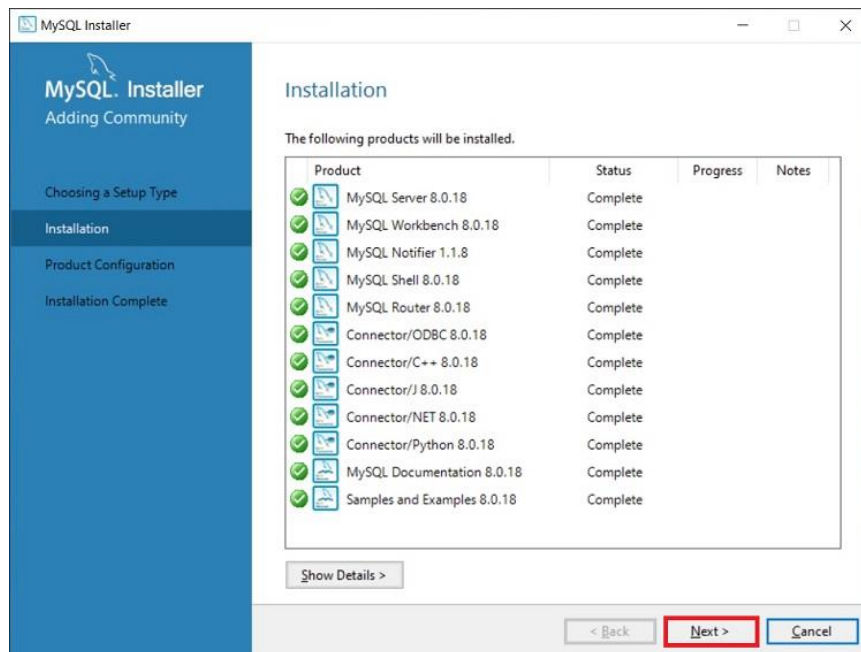


Рис. 49. Завершення встановлення компонентів MySQL

Крок 6 – Налаштування компонентів MySQL

Усі компоненти встановлено, але не всі налаштовані, тому нам необхідно їх налаштувати. Програма встановлення покаже, які конкретно компоненти потрібно налаштувати. У нашому випадку це MySQL Server, MySQL Router і Samples and Examples. Натискаємо «Next» (рис. 50).

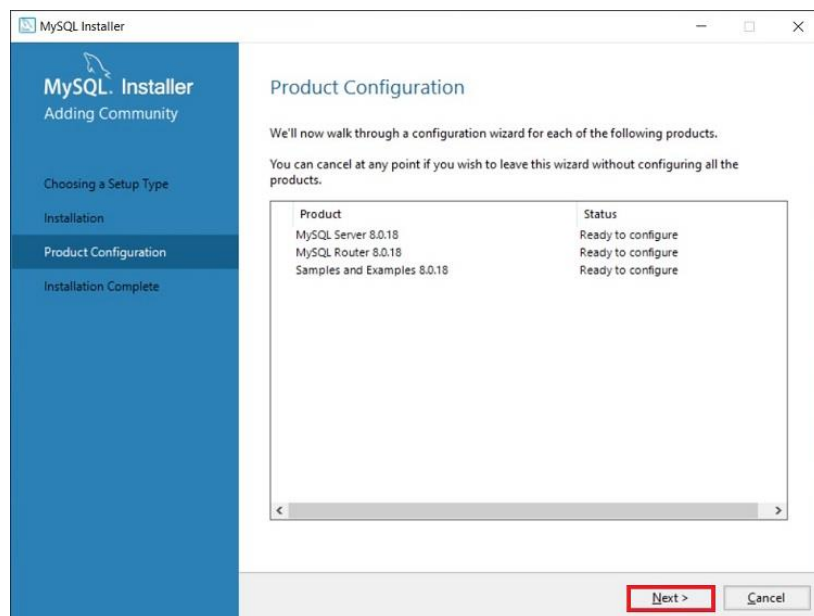


Рис. 50. Налаштування MySQL Server, MySQL Router і Samples and Examples

Крок 7 – Налаштування MySQL Server (параметр High Availability)

Спочатку обов'язково необхідно налаштувати MySQL Server. Тому налаштовуємо параметр «High Availability» (Висока доступність), який

відповідає за те, як буде встановлено MySQL сервер. У цьому випадку, як і в більшості інших, потрібен стандартний «Standalone MySQL Server» (Автономний сервер) – це класичний варіант встановлення MySQL Server. Тому вибрати перший пункт і натиснути «Next» (рис. 51).

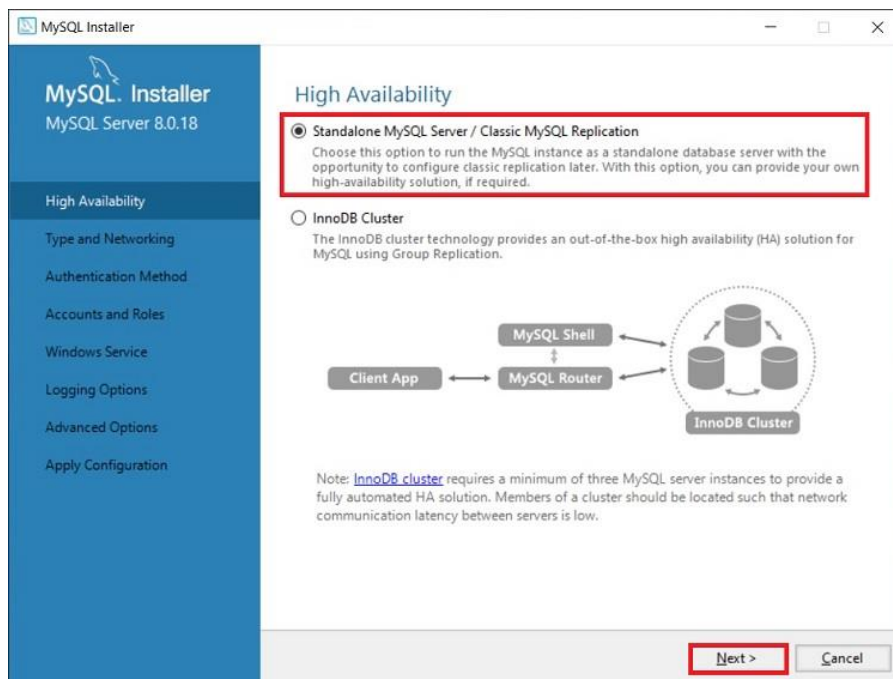


Рис. 51. Налаштування параметру «High Availability»

Крок 8 – Налаштування MySQL Server (Type and Networking)

Далі налаштовуємо тип сервера і мережу. У цьому разі можна залишити все за замовчуванням:

- тип – Development Computer;
- протокол – TCP/IP;
- порт – 3306;
- прапорець «Open Windows Firewall ports for network access» необхідно поставити.

У разі потреби можна більш тонко налаштувати MySQL Server, використавши для цього розширені параметри. Щоб це зробити, поставте галочку «Show Advanced and Logging Options». Для продовження тиснемо «Next» (рис. 52).

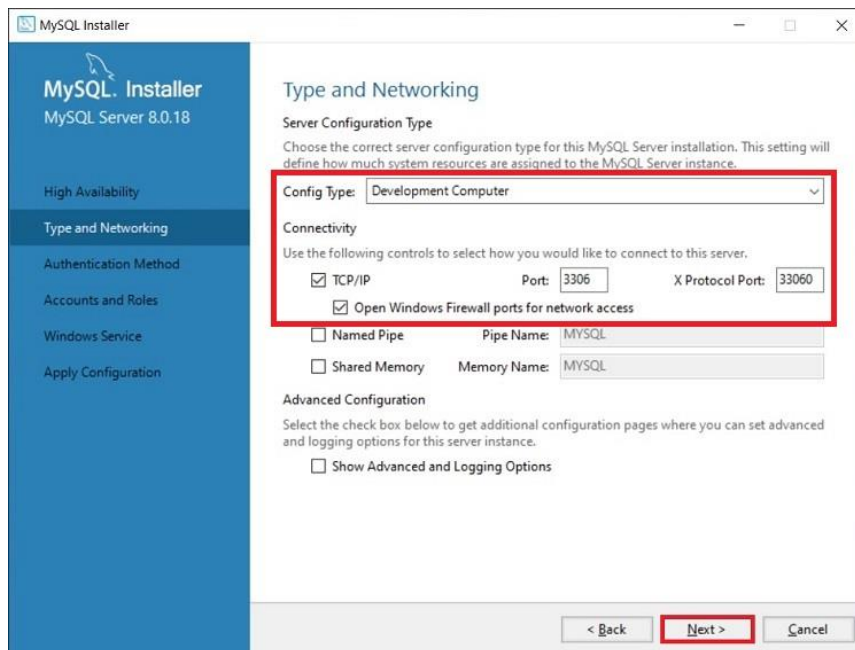


Рис. 52. Використання розширених параметрів

Крок 9 – Налаштування методу автентифікації в MySQL Server

8 версія MySQL підтримує новий метод автентифікації, який і рекомендовано використовувати, залишаємо як є і натискаємо «Next». У разі потреби можна вибрати другий пункт, який використовувався в MySQL 5 (рис. 53).

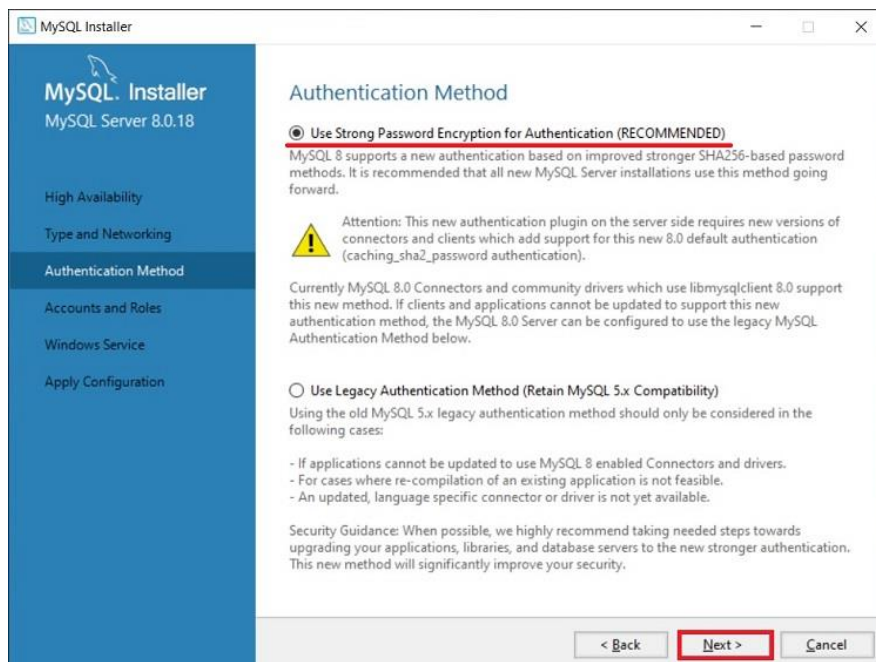


Рис. 53. Брання нового методу автентифікації при встановленні MySQL

Крок 10 – Налаштування користувачів MySQL

Після цього потрібно придумати пароль для користувача root (це головний адміністратор MySQL). Крім цього, можна додати додаткових

користувачів, щоб це зробити, необхідно натиснути на кнопку «Add User» (рис. 54).

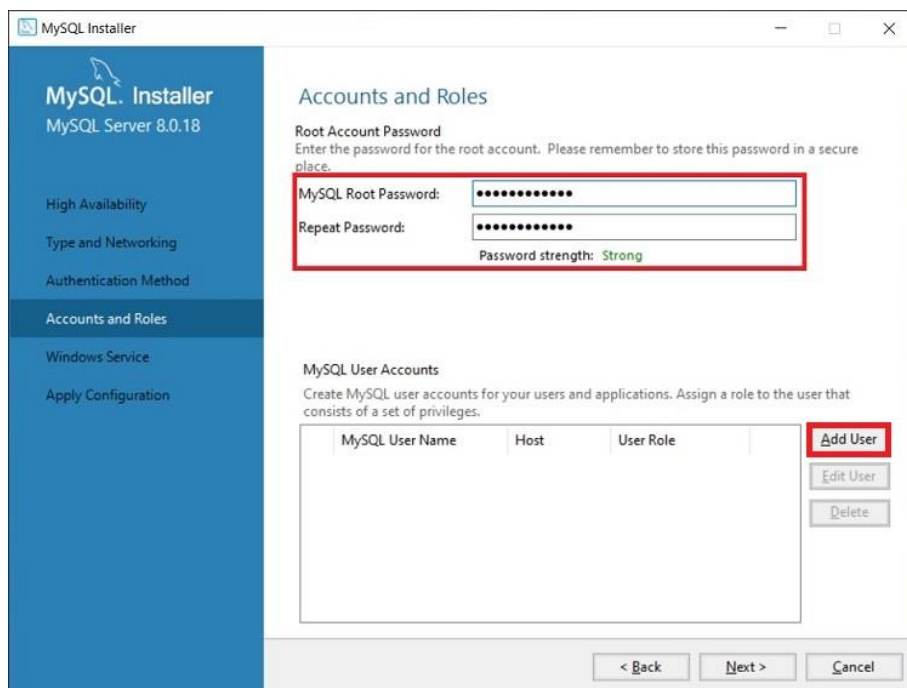


Рис. 54. Проведення авторизації

І ввести необхідні дані:

- User Name – вводим ім'я користувача;
- Host – залишаємо «All Hosts»;
- Role – обираємо роль користувача;
- Password та Confirm Password – придумуємо і вводим пароль, який буде у цього користувача. Натискаємо «OK» (рис. 55).

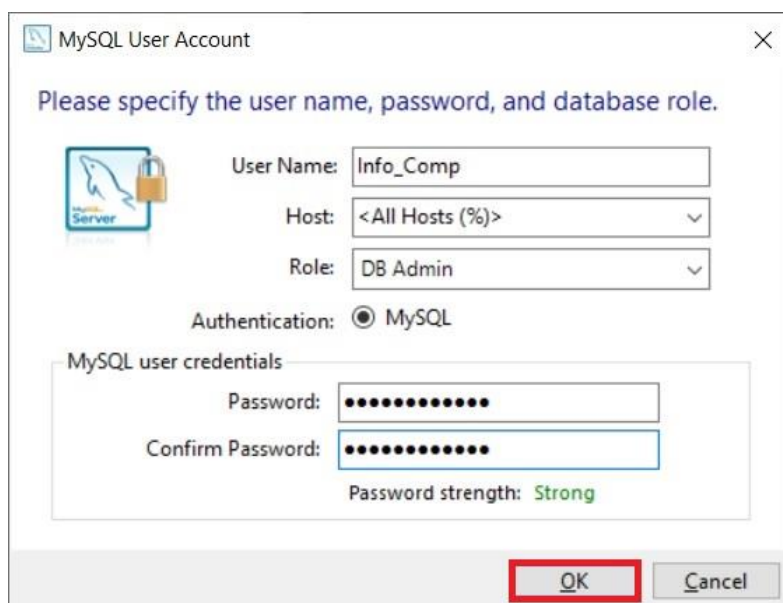


Рис. 55. Проведення авторизації (продовження)

Після цього користувача буде створено. Натискаємо «Next» для продовження (рис. 56).

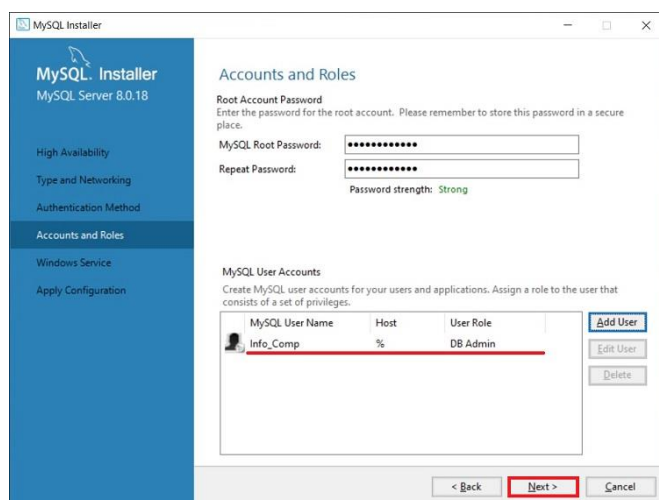


Рис. 56. Повідомлення про створення користувача

Крок 11 – Налаштування служби MySQL в Windows

Тепер нам необхідно налаштувати службу MySQL, яка працюватиме в Windows.

Можемо задати:

- Windows Service Name – Ім'я служби;
- «*Start the MySQL Server at System Startup*» – якщо ця галочка стоїть, то служба MySQL запускатиметься разом із запуском Windows;
- Run Windows Service – від імені якого облікового запису працюватиме служба MySQL в Windows;
- Standard System Account – це системний обліковий запис.

Можна все залишити за замовчуванням. Натискаємо «Next» (рис. 57).

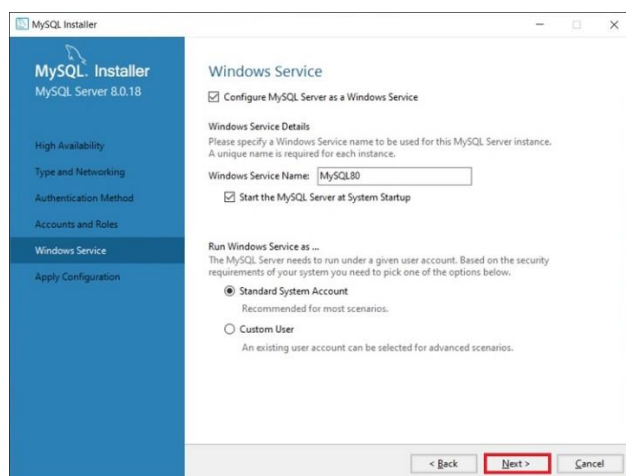


Рис. 57. Налаштування служби MySQL, яка працюватиме в Windows

Примітка. «Режим бога» в Windows - що це таке і як його ввімкнути в Windows 10?

Крок 12 – Застосування параметрів MySQL Server

Усі параметри MySQL Server налаштовані, однак нам ще потрібно їх застосувати та зберегти. Для цього на наступному вікні натискаємо «*Execute*» (рис. 58).

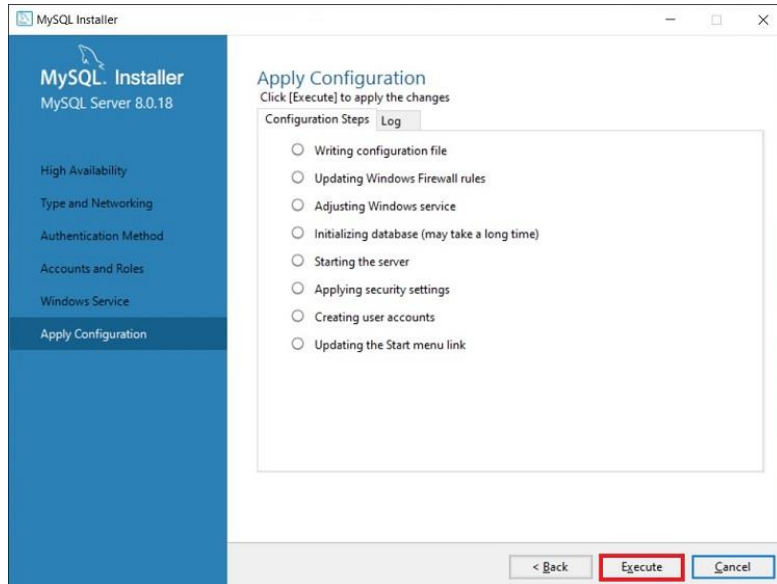


Рис. 54. Застосування та збереження налаштованих параметрів MySQL Server

Крок 13 – Завершення налаштування MySQL Server

Коли всі прапорці будуть проставлені і відобразиться повідомлення «Successful», процес буде завершено, натискаємо «*Finish*» (рис. 59).

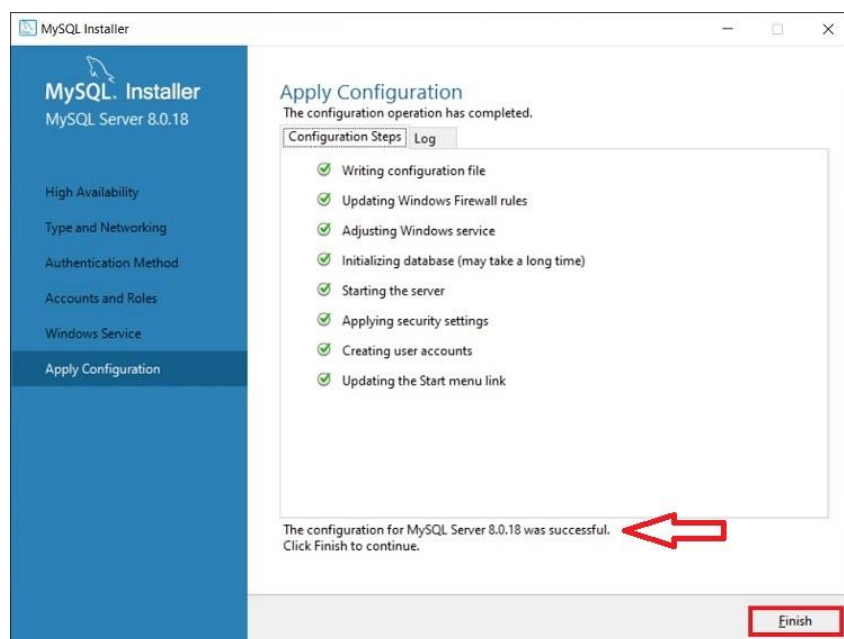


Рис. 59. Закінчення встановлення MySQL

Статус MySQL Server буде змінено на «Configuration complete». Щоб продовжити налаштування, натискаємо «*Next*» (рис. 60).

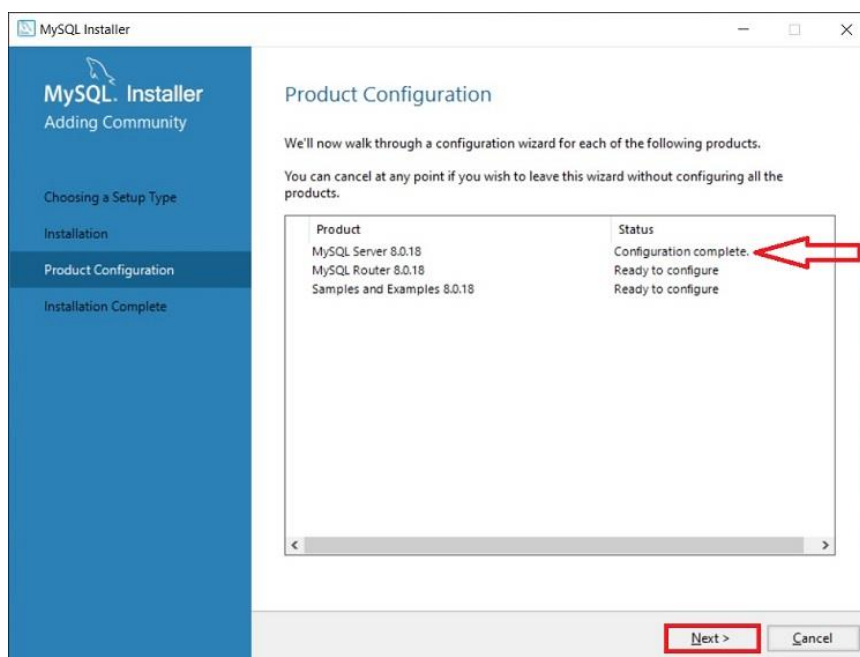


Рис. 60. Зміна статусу MySQL Server на «Configuration complete»

Крок 14 – Налаштування MySQL Router

У цьому випадку нам пропонують налаштувати маршрутизатор MySQL, однак це робити необов'язково, можемо відразу натиснути «*Finish*» (рис. 61).

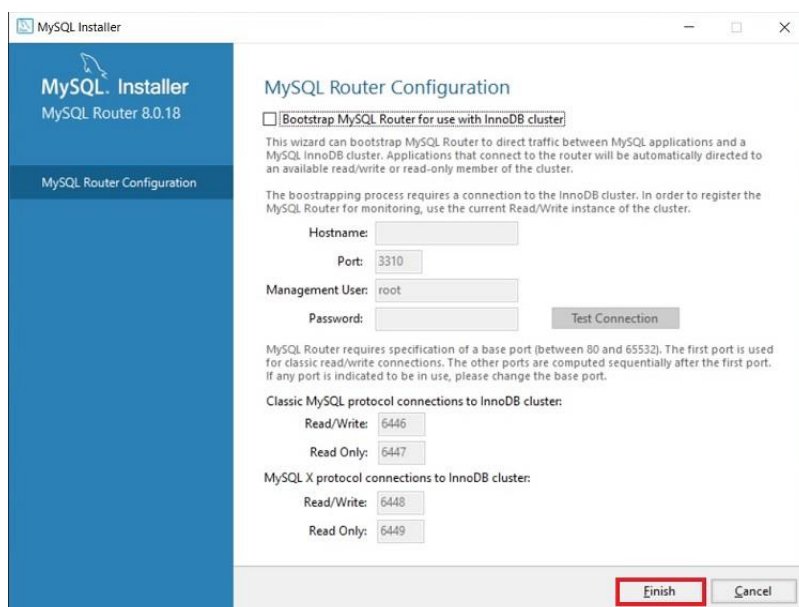


Рис. 61. Налаштування MySQL Router

Для того щоб перейти до налаштування наступного компонента, натискаємо «*Next*» (рис. 62).

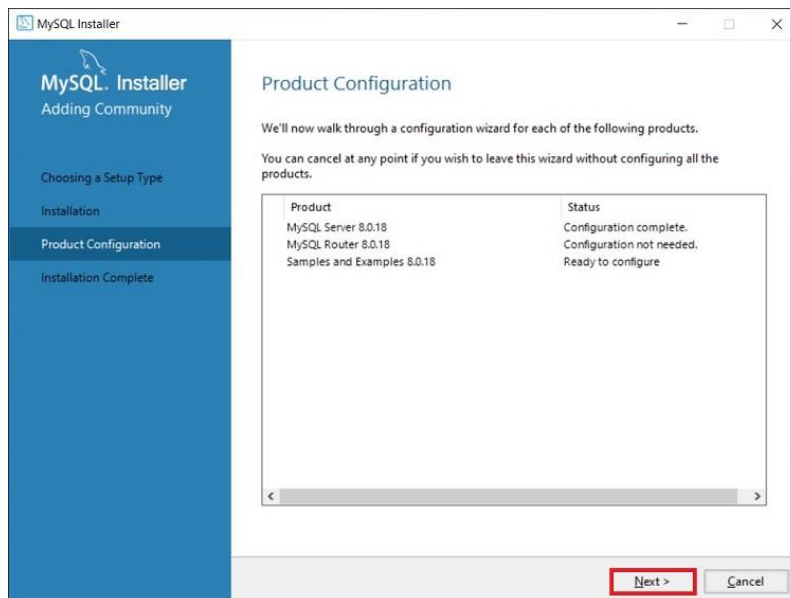


Рис. 62. Налаштування MySQL Router (закінчення)

Крок 15 – Встановлення тестових даних на MySQL

Далі ми можемо встановити тестові дані на MySQL Server, наприклад, для навчання. Щоб це зробити, вводимо пароль користувача Root і натискаємо «Check» для перевірки, якщо підключення встановлено, отже, все добре і можна продовжувати. Натискаємо «Next» (рис. 63).

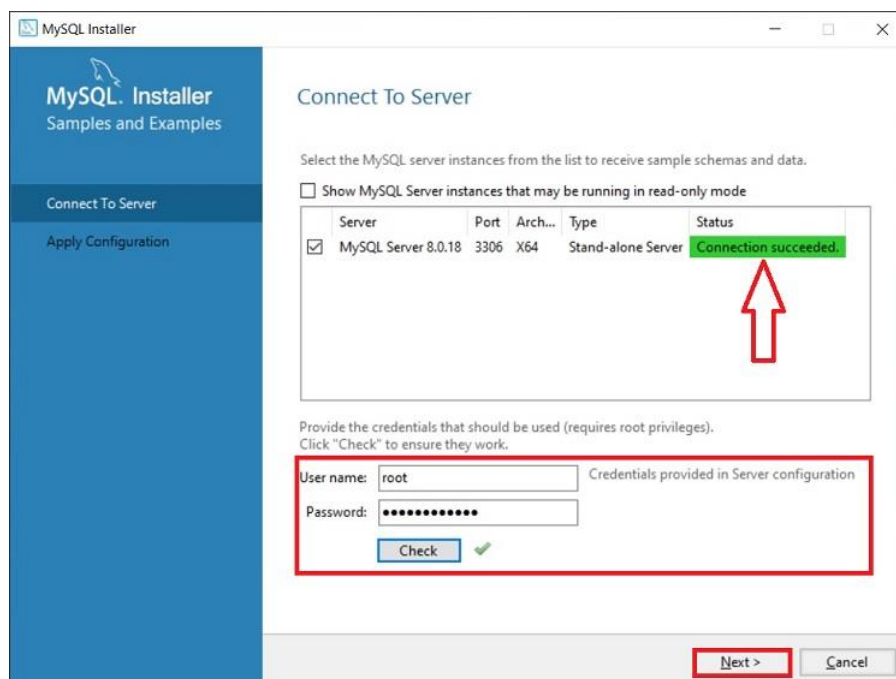


Рис. 63. Встановлення тестових даних на MySQL Server

Щоб застосувати всі параметри і почати витяг тестових даних на MySQL Server, натискаємо «Execute» (рис. 64).

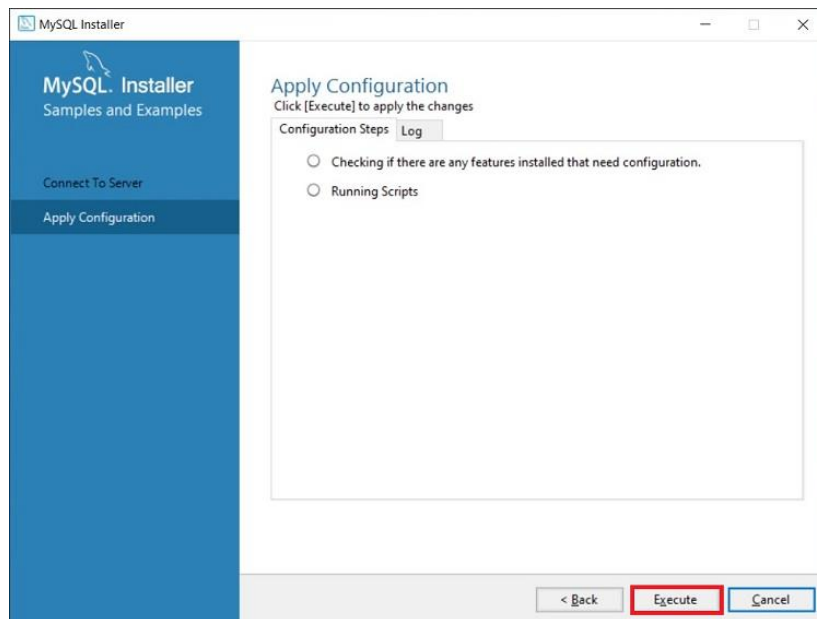


Рис. 64. Застосування всіх параметрів та початок витягування тестових даних на MySQL Server

Процес буде завершено, коли прапорці буде проставлено і відобразиться повідомлення «Successful». Натискаємо «*Finish*» (рис. 65).

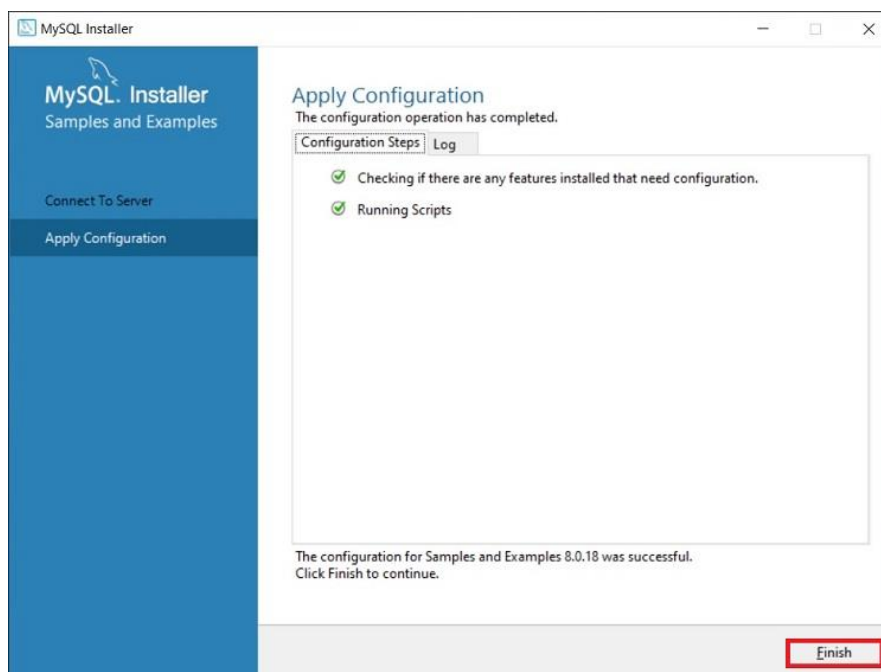


Рис. 65. Успішне завершення встановлення параметрів та тестування даних

Крок 16 – Завершення налаштування компонентів MySQL

Усі компоненти налаштовані. Натискаємо «Next» (рис. 66).

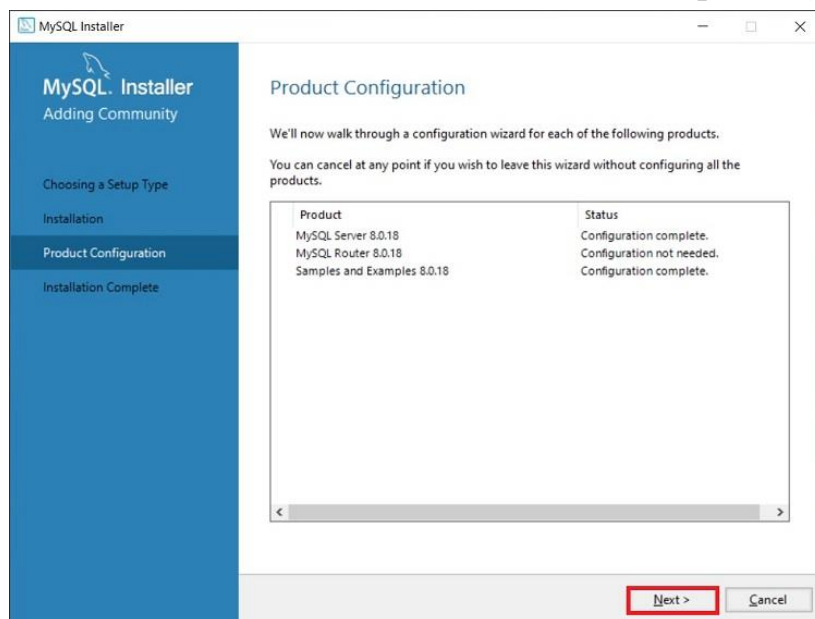


Рис. 66. Завершення налаштування компонентів MySQL

Крок 17 – Завершення всього процесу встановлення та налаштування MySQL

MySQL Server і всі необхідні для роботи з ним компоненти встановлені та налаштовані: це і MySQL Workbench, і різні конектори, і документація, і навіть тестова база даних.

Щоб відразу ж запустити MySQL Workbench після завершення встановлення, поставте галочку «Start MySQL Workbench after Setup». Натискаємо «Finish» (рис. 67).

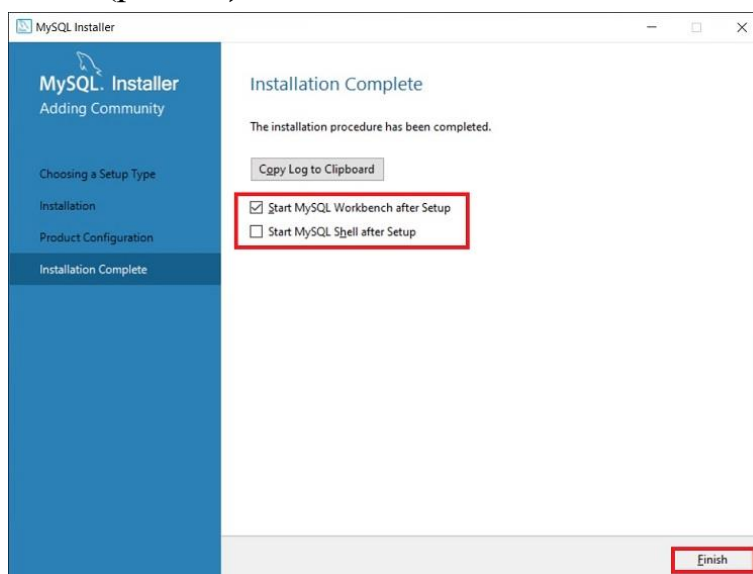


Рис. 67. Завершення всього процесу встановлення та налаштування MySQL

Примітки. Встановлення Microsoft SQL Server 2019 Express на Windows 10.

Запуск MySQL Workbench

Після встановлення MySQL у меню Windows 10 з'являться всі необхідні ярлики, які Ви можете використовувати. Наприклад, для запуску середовища «MySQL Workbench 8» можна використовувати відповідний ярлик (рис. 68).

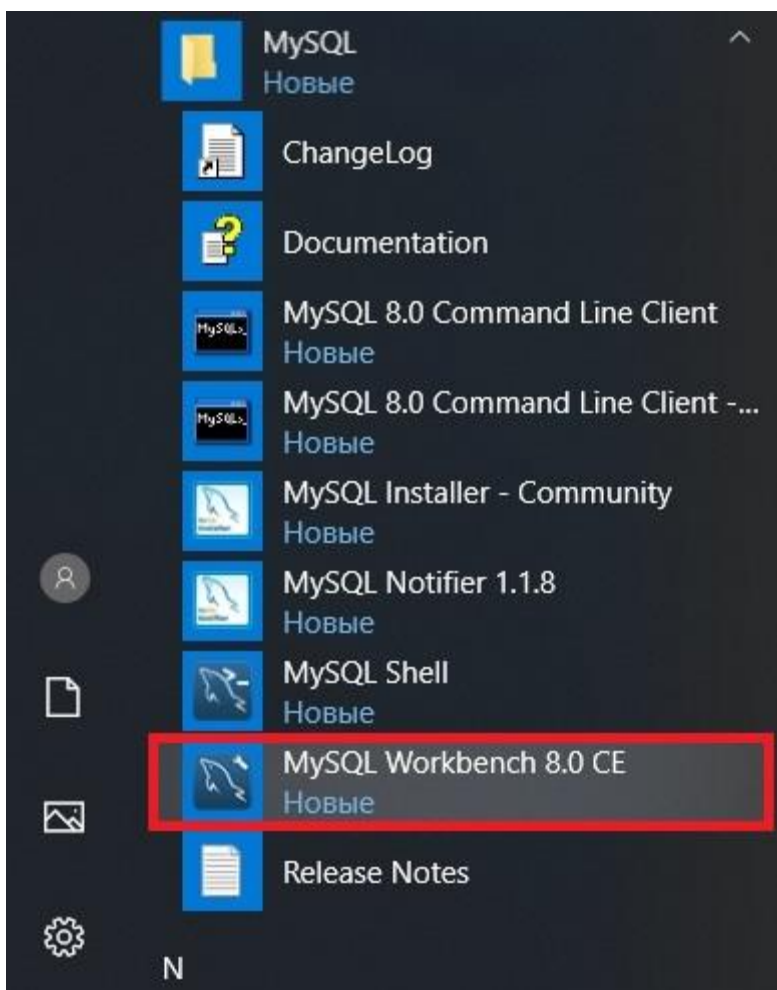


Рис. 68. Використання ярлику середовища «MySQL Workbench 8» для запуску

Примітки. Якщо Вас цікавить SQL і T-SQL, рекомендую пройти наші курси з T-SQL, за допомогою яких Ви «з нуля» навчитесь працювати з SQL і програмувати з використанням мови T-SQL в Microsoft SQL Server.

Підключення до MySQL використовуючи MySQL Workbench

Щоб підключитися до сервера MySQL і подивитися на об'єкти бази даних, на таблиці, подання тощо, запустіть MySQL Workbench, наприклад, як зазначено вище. У результаті відкриється MySQL Workbench, де на стартовій сторінці вже буде одне підключення до локального сервера, яке здійснюватиметься від імені користувача **root**. Натискаємо на нього (рис. 69).

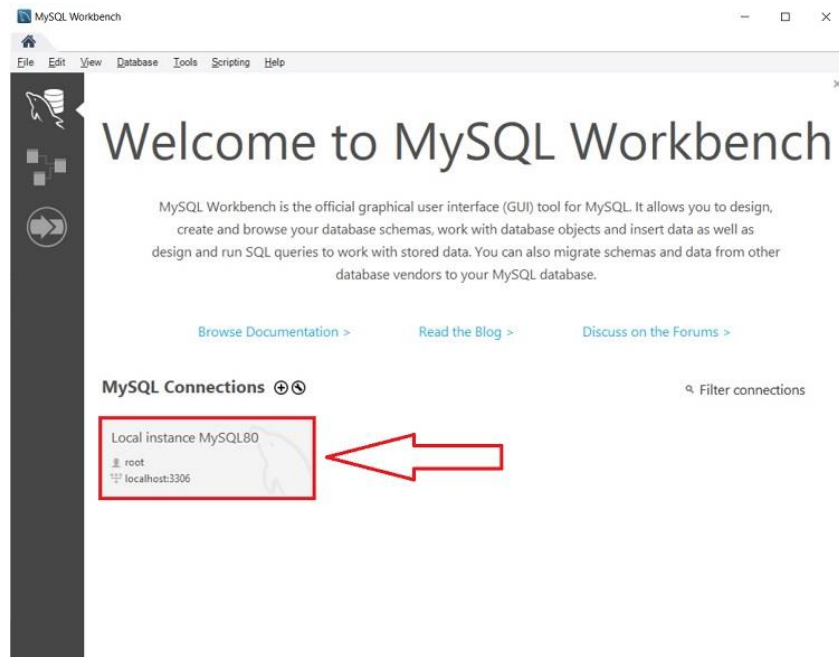


Рис. 69. Початок підключення до MySQL

Потім вводимо пароль, щоб зберегти його і не вводити щоразу під час підключення, ми можемо поставити прапорець «**Save password in vault**». Натискаємо «**OK**» (рис. 70).

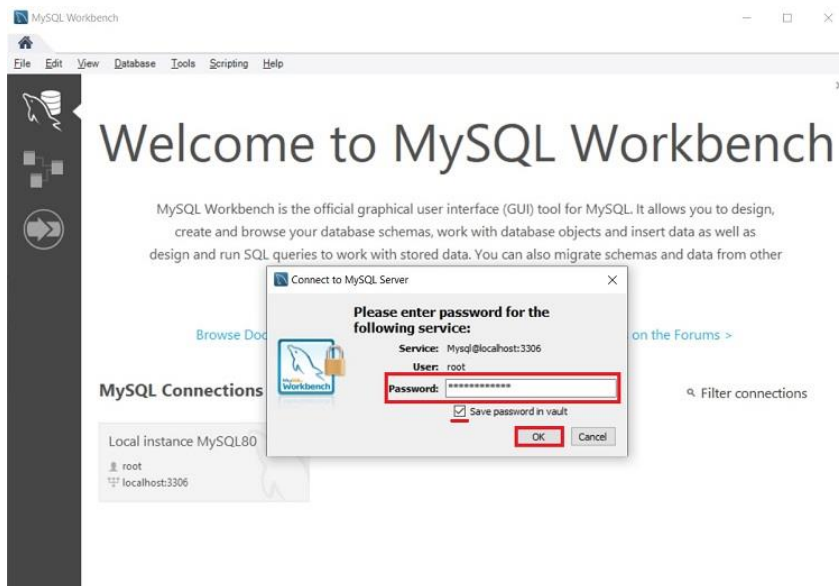


Рис. 70. Введення пароля

У підсумку підключимося до сервера, і щоб подивитися на схему даних, нам у вікні навігатора необхідно перейти на вкладку «**Schemas**». Крім того, можна приховати непотрібні вікна, для цього можна використовувати спеціальні значки в правій верхній частині середовища **MySQL Workbench** (рис. 71).

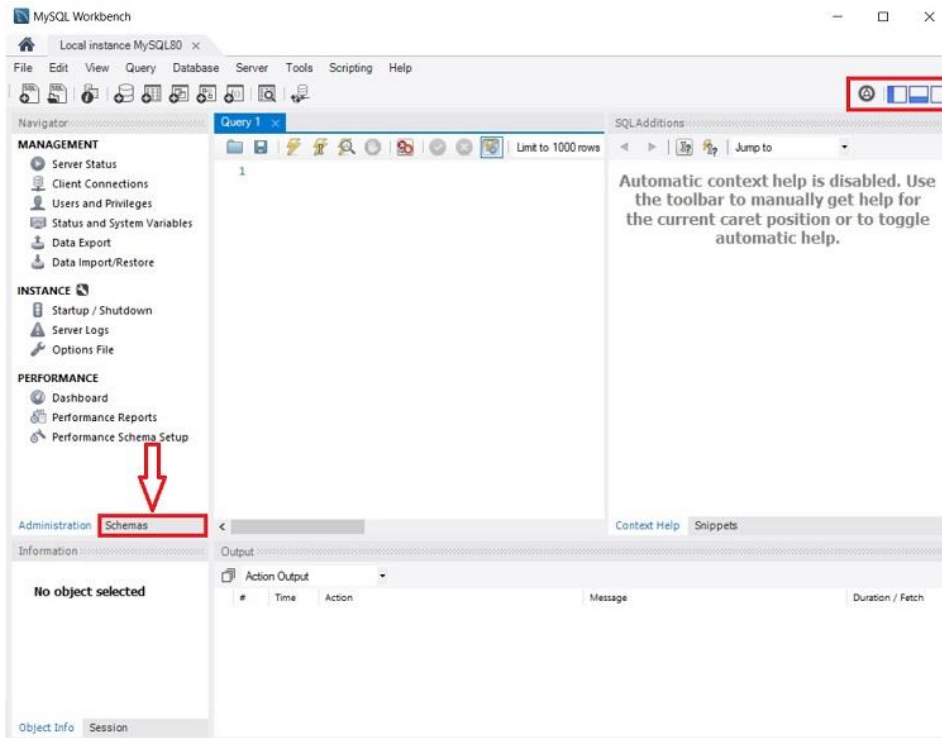


Рис. 71. Підключення до схеми даних

Відобразиться схема даних, тепер можна подивитися, які об'єкти є в базі даних і, наприклад, вчитися писати SQL запити. Для наочності приховано бічне вікно (рис. 72).

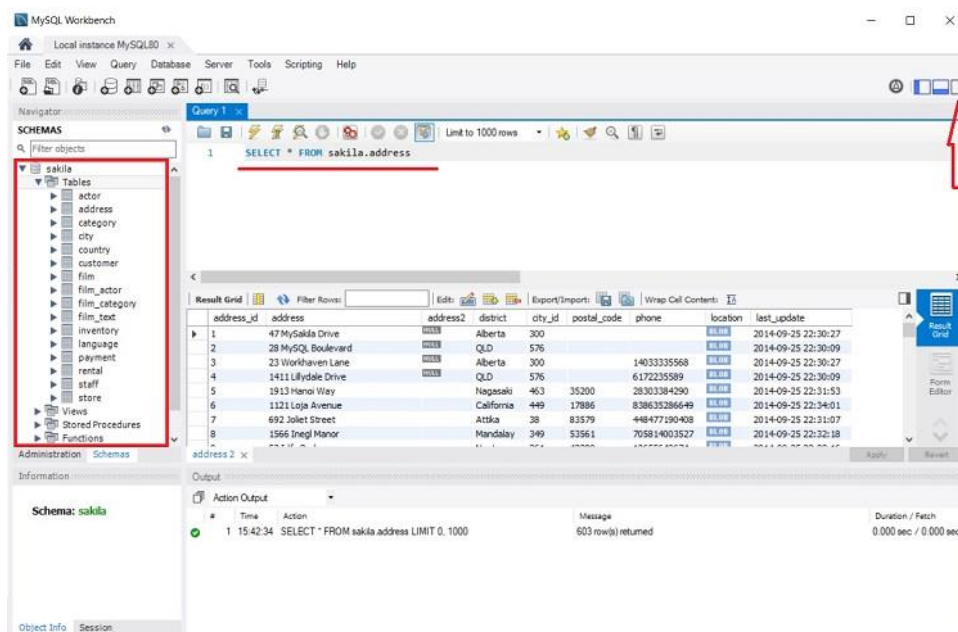


Рис. 72. Відображення схеми даних

У цьому прикладі було надіслано запит **SELECT** на вибірку даних із таблиці **address**. Можна спостерігати як дані повернулися і все працює.

Виконання завдання 2

Завантаження, встановлення, налаштування та розгортання СУБД PostgreSQL (SQL Shell) на Windows 10.

PostgreSQL (або просто Postgres) – це об'єктно-реляційна система управління базами даних (СУБД), відома своєю надійністю, потужністю та гнучкістю.

Щоб використовувати PostgreSQL, необхідно встановити:

- 1) сервер бази даних PostgreSQL;
- 2) графічний інструмент для адміністрування та управління БД. pgAdmin – найпопулярніший інструмент із графічним інтерфейсом для PostgreSQL.

Крок 1) Відкрити браузер.

Перейти на <https://www.postgresql.org/download> та Windows (рис. 73).

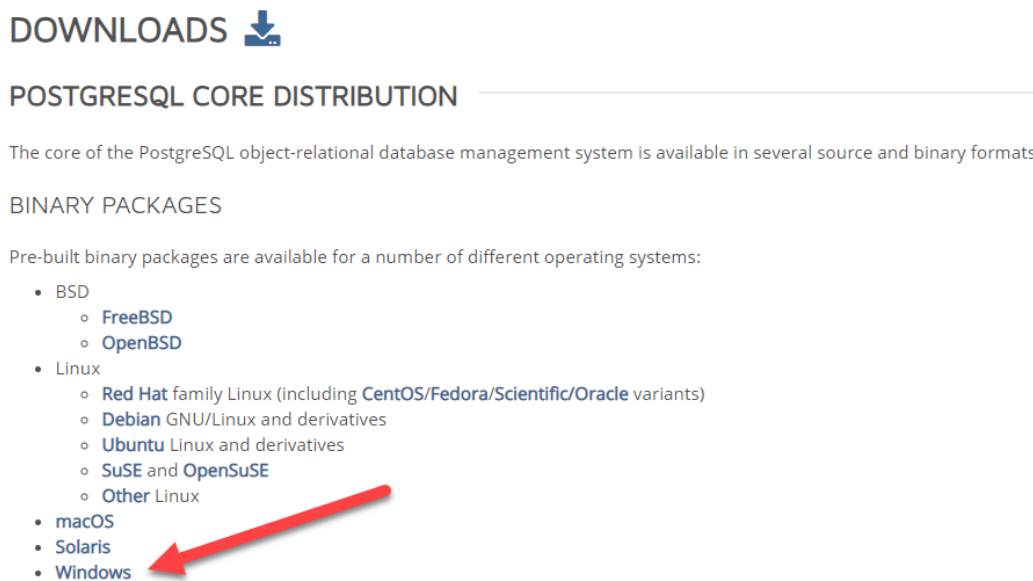


Рис. 73. Обрання операційної системи для встановлення СУБД Postgre SQL

Крок 2) Перевірити варіанти.

Надається два варіанти:

- 1) інтерактивний установник від EnterpriseDB;
- 2) графічний установник від BigSQL.

BigSQL наразі встановлює pgAdmin версії 3, яка застаріла. Найкраще вибрати EnterpriseDB, який встановлює останню версію 4 (рис. 74).

WINDOWS INSTALLERS

INTERACTIVE INSTALLER BY ENTERPRISEDB

Download the installer certified by EnterpriseDB for all supported PostgreSQL versions.

This installer includes the PostgreSQL server, pgAdmin; a graphical tool for managing and developing your databases, and StackBuilder; a package manager used to download and install additional PostgreSQL tools and drivers. Stackbuilder includes management, integration, migration, replication, geospatial, and other tools.

This installer can run in graphical or silent install mode.

The installer is designed to be a straightforward, fast way to get up and running with PostgreSQL on Windows.

Advanced users can also download a **zip archive** of the binaries, without the installer. This download is intended for users who wish to include PostgreSQL in another application installer.

PLATFORM SUPPORT

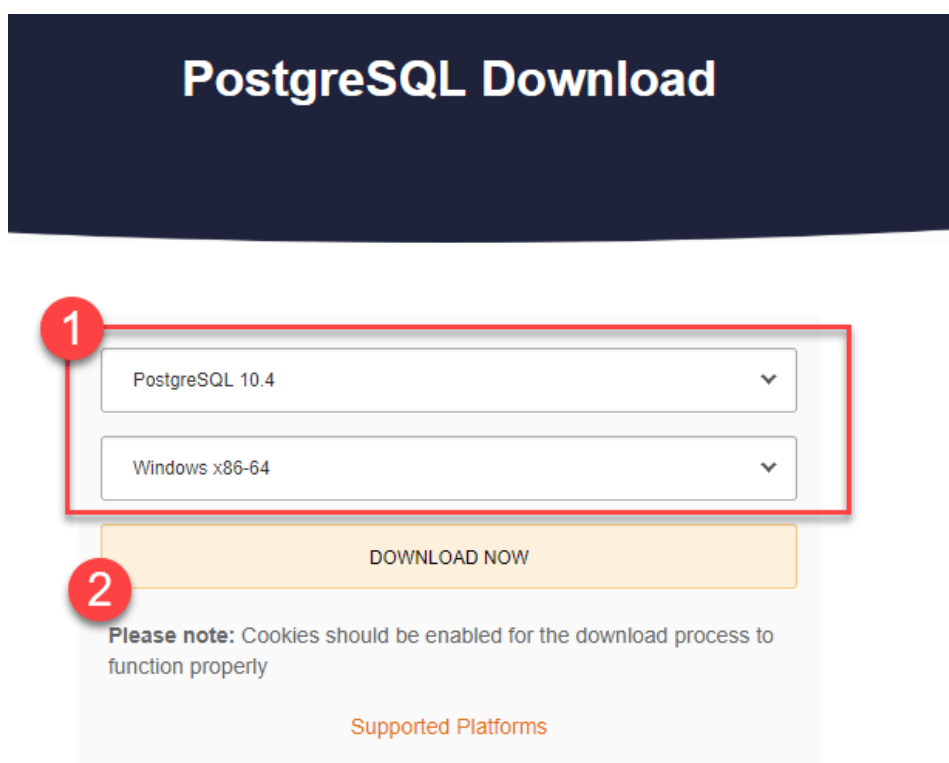
The installers are tested by EnterpriseDB on the following platforms. They can generally be expected to run on other comparable versions:

PostgreSQL Version	64 Bit Windows Platforms	32 Bit Windows Platforms
10	2016, 2012 R2 & R1, 2008 R2, 7, 8, 10	2008 R1, 7, 8, 10

Рис. 74. Обрання параметру EnterpriseDB, який встановлює останню версію pgAdmin 4

Крок 3) Виберіть PostgreSQL версія.

1. Буде запропоновано бажану PostgreSQL версію. Обрати останню PostgreSQL версію та ОС відповідно до необхідного середовища.
2. Натиснути кнопку **DOWNLOAD** (рис. 75).



PostgreSQL Download

1

PostgreSQL 10.4

Windows x86-64

2

DOWNLOAD NOW

Please note: Cookies should be enabled for the download process to function properly

[Supported Platforms](#)

Рис. 75. Обрання версії PostgreSQL та операційної системи

Крок 4) Відкрити ехе-файл.

Щойно завантажиться PostgreSQL, відкрити завантажений ехе-файл і натиснути Далі на екрані привітання встановлення (рис. 76).

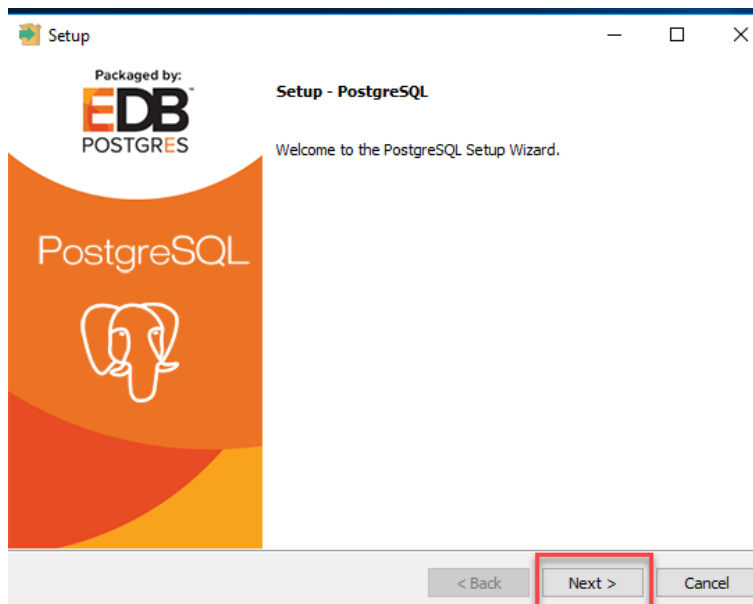


Рис. 76. Відкриття ехе-файлу

Крок 5) Оновити місце розташування.

1. За необхідності змініть каталог встановлення, інакше залиште його за замовчуванням.
2. Натиснути **Далі** (рис. 77).

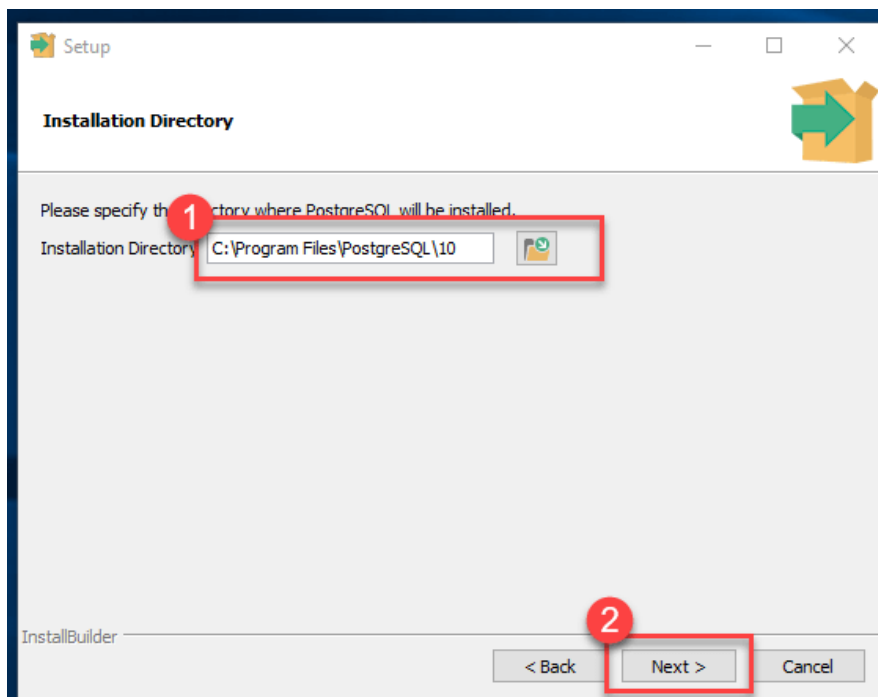


Рис. 73. Оновлення місця розташування

Крок 6) Вибрати компоненти.

1. Можна вибрати компоненти, які необхідно встановити у свою систему. Можна зняти прапорець **Stack Builder**.
2. Натиснути **Далі** (рис. 78).

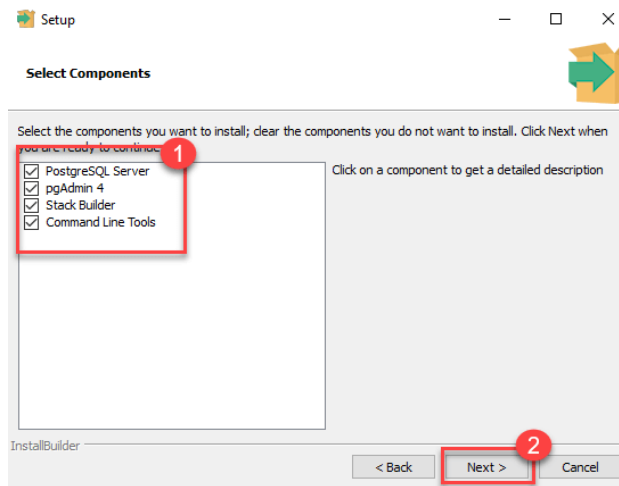


Рис. 78. Обрання компонентів

Крок 7) Перевірити розташування даних.

1. Можна змінити місце розташування даних.
2. Натиснути Далі (рис. 79).

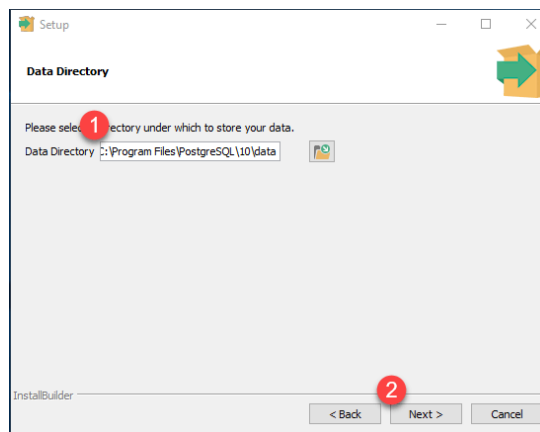


Рис. 79. Перевірка розташування даних

Крок 8) Ввести пароль.

1. Ввести пароль суперкористувача. Записати це.
2. Натиснути Далі (рис. 80).

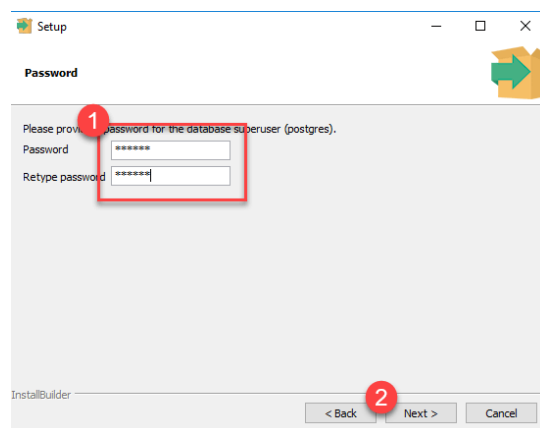


Рис. 80. Внесення паролю користувача

Крок 9) Перевірити опцію порту.

1. Залишити номер порту за замовчуванням.
2. Натиснути Далі (рис. 81).

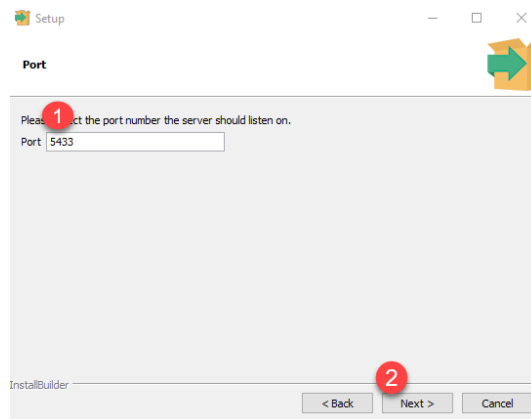


Рис. 81. Перевірка опції порту

3. Обрати **local** (рис. 82).

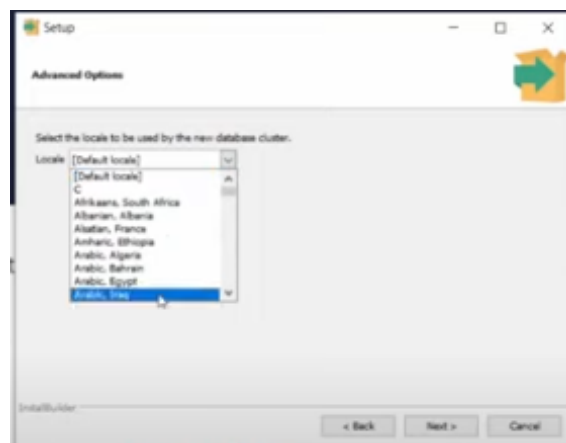


Рис. 82. Обрання режиму **local**

Крок 10) Перевірити зведення.

1. Перевірити зведення перед установкою.
2. Натиснути Далі (рис. 83).

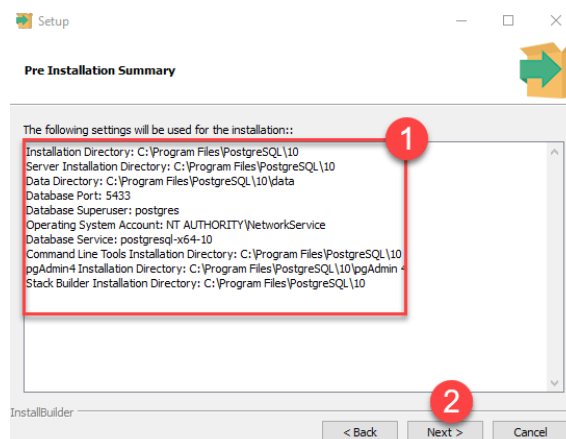


Рис. 83. Перевірка зведень

Крок 11) Готовий до встановлення.

Натиснути кнопку Next (рис. 84).

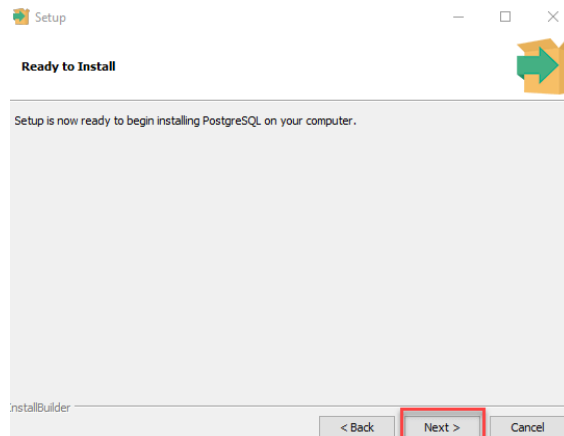


Рис. 84. Готовий до встановлення

Крок 12) Перевірити підказку побудовника стека.

Після завершення встановлення з'явиться запрошення **Stack Builder**.

1. Зняти прапорець із цієї опції. Використовувати **Stack Builder** у більш просунутих посібниках.
2. Натиснути **Finish** (рис. 85).

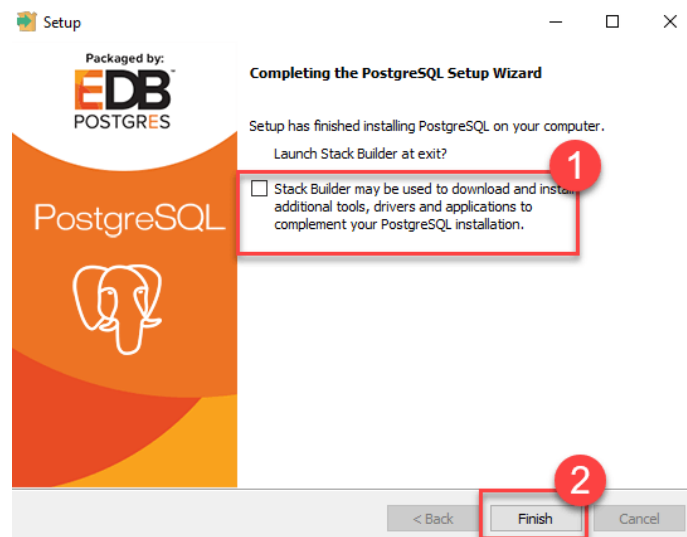


Рис. 85. Перевірка підказок побудовника стека

Виконання завдання 3

Завантаження, встановлення, налаштування та розгортання СУБД
PostgreSQL (pgAdmin) на Windows 10

Крок 1) Обрання локального серверу (рис. 86).

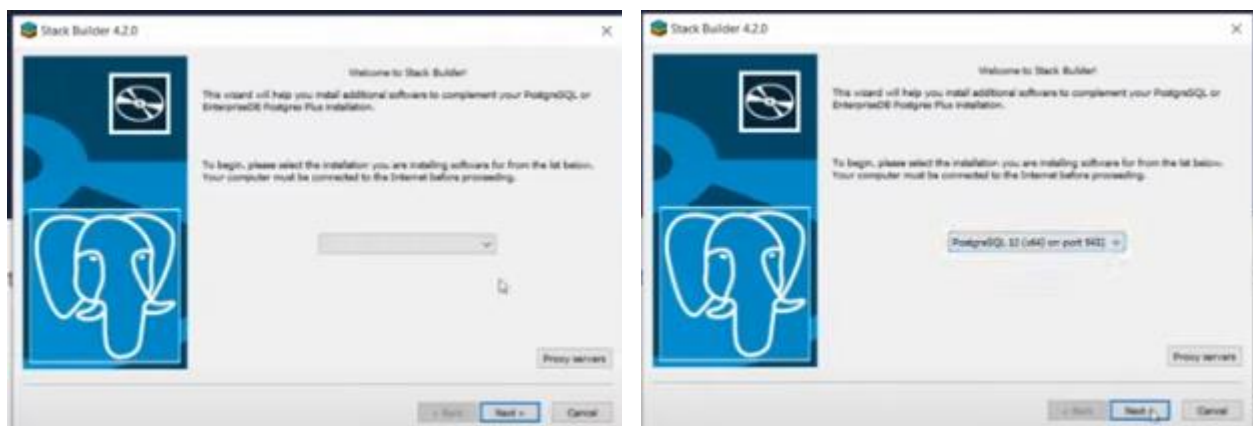


Рис. 86. Обрання локального серверу

Завантажуються **Stack Builder** для подальшого підключення необхідних плагінів та закриваємо. Після чого проводимо запуск (рис. 87).



Рис. 87. Запуск встановлених плагінів

Крок 2) Презентація PostgreSQL.

Запустити PostgreSQL, перейти в меню **Пуск** і знайти pgAdmin (рис. 88).

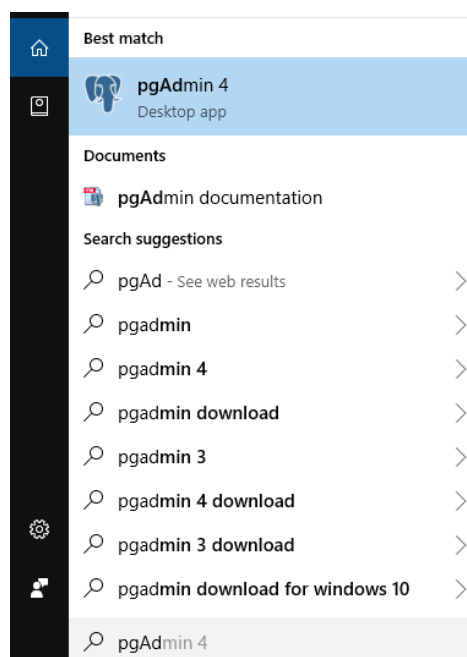


Рис. 88. Запуск PostgreSQL

Крок 3) Перевірити pgAdmin.

Ввести домашню сторінку pgAdmin (рис. 89).

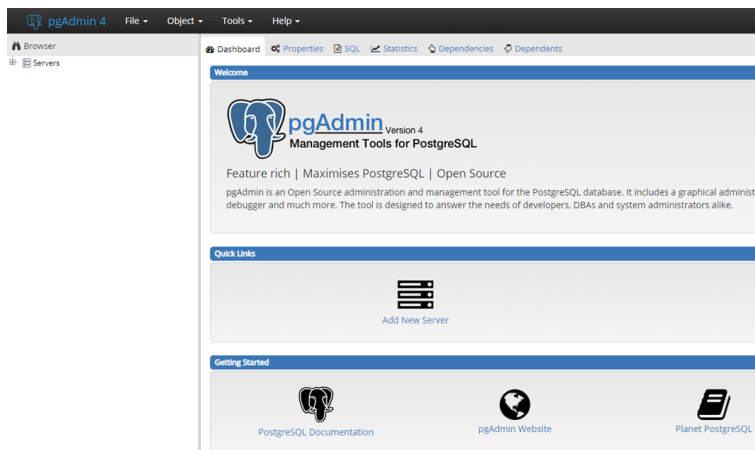


Рис. 89. Введення домашньої сторінки pgAdmin

Крок 4) Знайдіть PostgreSQL 10.

Натисніть Сервери > PostgreSQL 10 у лівому дереві (рис. 90).

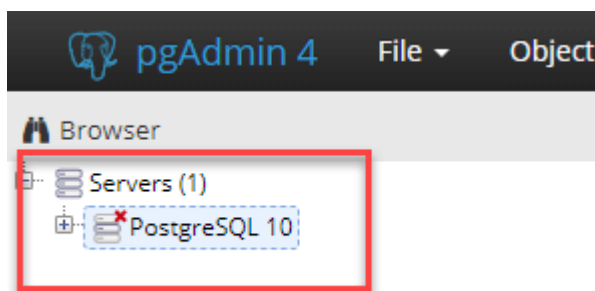


Рис. 90. Обрання серверу PostgreSQL 10

Крок 5) Ввести пароль.

1. Ввести пароль суперкористувача, створений під час встановлення.
2. Натиснути **ОК** (рис. 91).

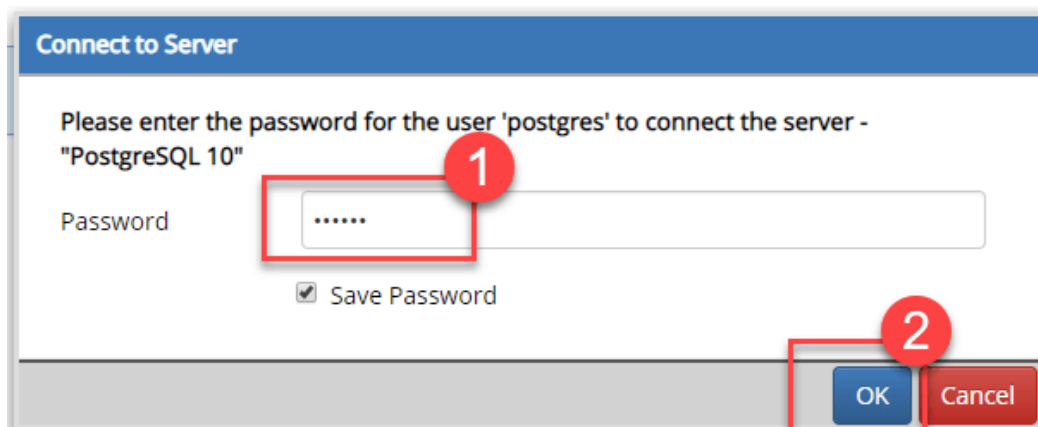


Рис. 91. Введення паролю суперкористувача

Крок 6) Перевірити панель моніторингу.
Представляється панель управління (рис. 92).

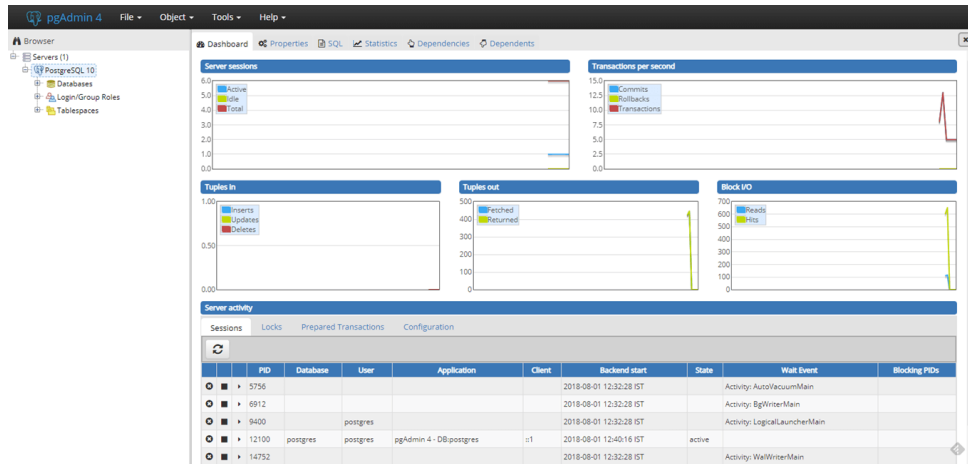


Рис. 92. Панель моніторингу

Контрольні запитання та завдання

1. Які особливості редакції MySQL Standard Edition?
2. Які особливості редакції MySQL Cluster (CGE)?
3. Які особливості редакції MySQL Community Edition?
4. У чому полягає різниця між СУБД MySQL та СУБД PostgreSQL?
5. Визначте особливості СУБД MySQL.
6. Визначте особливості СУБД PostgreSQL.
7. Що відноситься до недоліків використання СУБД MySQL?
8. Що відноситься до недоліків використання СУБД PostgreSQL?

ЛАБОРАТОРНА РОБОТА № 3

ВІЗУАЛЬНЕ ПРОЄКТУВАННЯ БАЗИ ДАНИХ В MYSQL WORKBENCH ТА В POSTGRESQL

Мета:

проєктування та створення бази даних за допомогою інструменту для візуального проєктування баз даних MySQL Workbench від компанії Oracle, отримання ER-моделі і SQL-дампу, а також за допомогою SQL SHELL (консолі) та pgAdmin 4.

Завдання:

1. Створити таблиці бази даних та визначити зв'язки між ними за допомогою MySQL Workbench.
2. Створити таблиці бази даних та визначити зв'язки між ними за допомогою PostgreSQL (SQL Shell).
3. Створити таблиці бази даних та визначити зв'язки між ними за допомогою PostgreSQL (pgAdmin).

Виконання завдання 1

Створення таблиці бази даних та визначити зв'язки між ними за допомогою MySQL Workbench

Зовнішній вигляд вікна програми, розділ «Моделювання даних» має такий вигляд (рис. 93).



Рис. 93. Вікно програми – розділ «Моделювання даних»

Для того, щоб відкрити наявну модель, треба натиснути на посилання: **Open Existing EER Model**, для створення нової моделі – вибрати параметр: **Create New EER Model**, щоб створити модель «сутність-зв'язок» з наявної бази даних, – натиснути на параметр: **Create EER Model From Existing Database**, а для створення EER моделі із SQL скрипта треба вибрати: **Create EER Model From SQL Script**. Для створення нової моделі скористаємося посиланням **Create New EER Model**, після натискання на нього відобразиться вікно з параметрами (рис. 94).

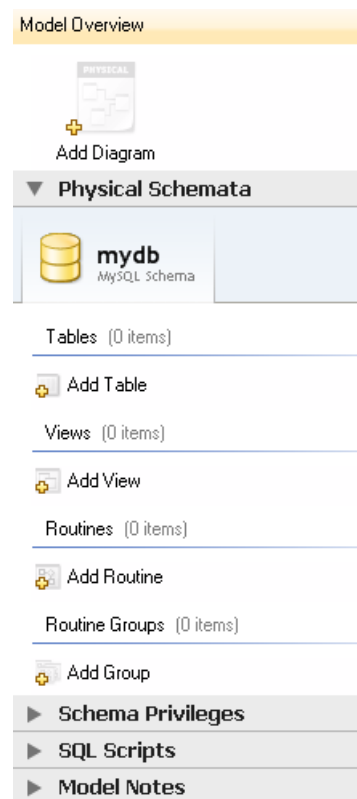


Рис. 94. Відображення вікна з параметрами

На початку, необхідно створити таблиці, для цього натиснемо на кнопку **Add Table**, з'явиться така форма (рис. 95).

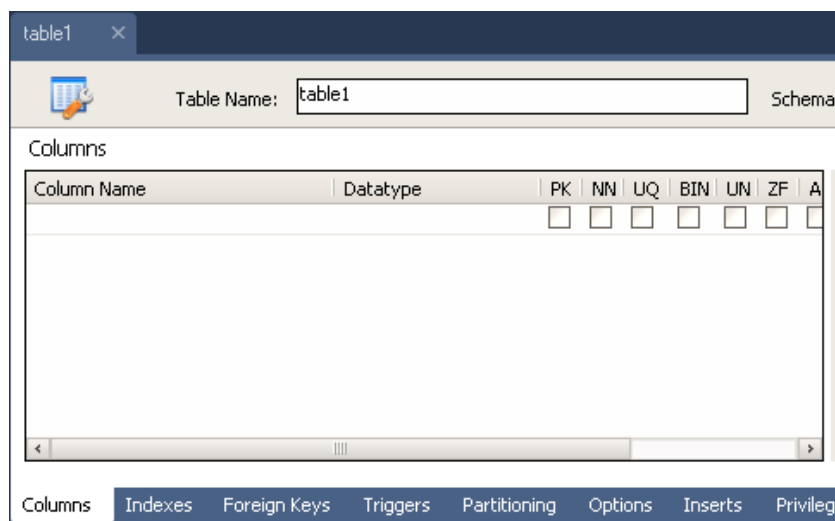


Рис. 95. Створення таблиць

Спочатку створимо таблицю **users**, яка зберігатиме дані про користувачів інформаційної системи, у полі **table Name** впишемо ім'я таблиці, у розділі форми **Columns** створимо поля таблиці:

- перше поле **id** міститиме унікальний номер користувача, задамо йому властивості: **Auto Increment**, **Not Null**, **Primary key** і **Unique**, у розділі **Data type** виберемо цілочисельний тип **integer**;

- друге поле **fio**, де зберігатиметься **П. І. Б.** користувача, встановимо полю властивості: **Not Null**, **Primary key**, у розділі **Data type** виберемо строковий тип **VARCHAR** і задамо кількість символів у **255**;

- третє поле **login** міститиме логін користувача, воно має бути унікальним, як і поле **id**, тому встановимо йому властивість **Unique** і задамо кількість символів у **255**;

- такі поля: **password**, що містить пароль, **e_mail**, що містить адресу електронної пошти, і поле **type**, що містить тип користувача, будуть без особливих властивостей, зі строковим типом **VARCHAR** довжиною в **255** символів, за винятком останнього поля **type**, якому вистачить з **45** символів.

Після виконаних маніпуляцій форма з ім'ям таблиці **users** матиме такий вигляд, як наведено на рис. 96.

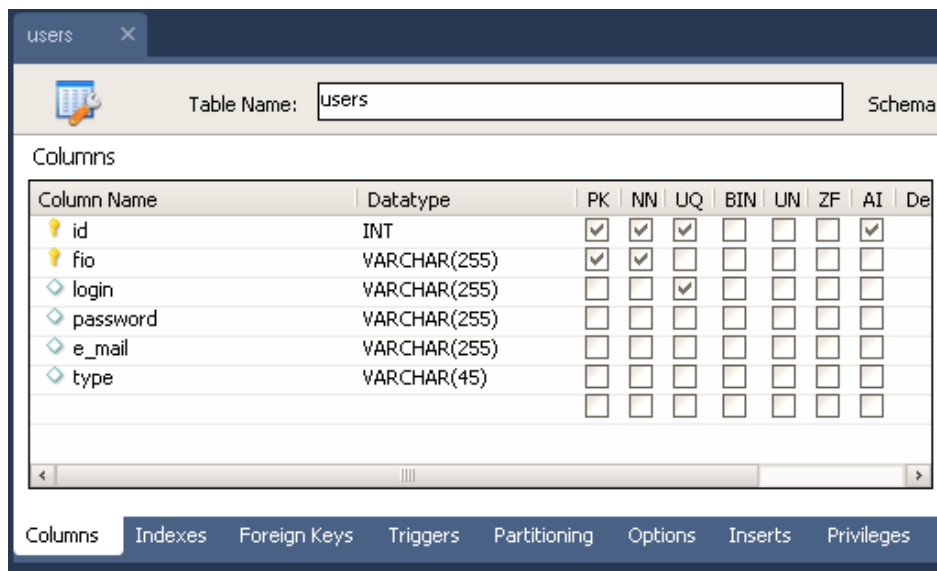


Рис. 96. Створення таблиці **users**

На діаграмі з'явиться таблиця **users** з полями та індексами (рис. 97).

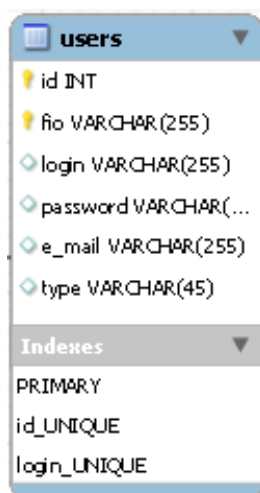


Рис. 99. Поля таблиці **users**

Аналогічним способом створимо таблицю **settings** з налаштуваннями доступу до бази даних інформаційної системи, що містить поля **id**, **host** для вказівки імені хоста (адреси сервера), **db** – імені бази даних, **user** і **password** з ім'ям користувача і паролем, для встановлення інформаційної системи на віддалений сервер.

Далі за вже відомим методом створимо таблицю **shops**, яка зберігатиме дані про магазини в полях: **id** типу **integer** – ключове, ненульове, унікальне з автоінкрементом, поле **name**, яке зберігає назву магазину, поле **address** – його фізичну адресу, поле **tel** – телефон магазину, **site** – інтернет-сайт магазину і поле **email** з електронною адресою магазину.

Потім створимо таблицю **products**, що зберігає дані про продукцію магазину в полях: **id** типу **integer** – ключове, ненульове, унікальне з автоінкрементом, поле **name**, що зберігає назву магазину, ключове, ненульове поле цілочисельного типу **shop_id**, що зберігає номер магазину, поле **type_id** з інформацією про номер товару з таблиці видів продукції. Поле **brand** – бренд виробника довжиною в 255 символів, поле **model** – з моделлю товару, поле **data** – з даними і характеристиками товару типу **Tinytext**, поле **img** з повною адресою до зображення товару довжиною в 255 символів, і поля **price** з ціною товару і **warranty** з інформацією про терміни гарантії на товар довжиною в 45 символів.

Створені таблиці **settings**, **shops** і **products** мають такий вигляд (рис. 98).

settings	Shops	Products
id INT	id INT	id INT
host VARCHAR(45)	name VARCHAR(255)	shop_id INT
db VARCHAR(45)	address VARCHAR(255)	type_id INT
user VARCHAR(45)	tel VARCHAR(100)	brand VARCHAR(255)
password VARCHAR(45)	site VARCHAR(100)	model VARCHAR(255)
	email VARCHAR(100)	data TINYTEXT
		img VARCHAR(255)
		price VARCHAR(45)
		warranty VARCHAR(45)
Indexes	Indexes	Indexes
PRIMARY	PRIMARY	PRIMARY
id_UNIQUE	id_UNIQUE	

Рис. 98. Склад таблиць **settings**, **shops** і **products**

Далі нам знадобиться таблиця, що зберігає тип продуктів **product_type**, вона складається з унікального, ненульового, ключового поля **id** з автоінкрементом цілочисельного типу й унікального поля **name** довжиною 255 символів, яке містить назву виду продуктів (рис. 99).

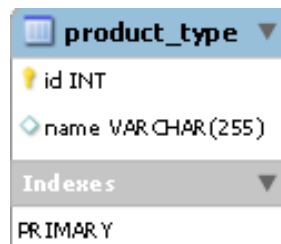


Рис. 99. Створення таблиці **product_type**

Останні дві таблиці це **orders** і **deliveries**, перша містить інформацію про замовлення клієнтів, а остання дані про доставку продукції.

Поля таблиці **orders**: **id** ключове, ненульове, унікальне поле цілочисельного типу з автоінкрементом, поле **shop_id**, яке містить номер магазину, – ключове, ненульове цілочисельного типу, поле **product_id**, яке зберігає номер продукту, – ключове, ненульове цілочисельного типу, поле **fio** з номером користувача, який здійснив замовлення, – ключове, ненульове цілочисельного типу, поле **date** с датою замовлення – типу **DATE**, поле **quantity** з кількістю замовлених товарів – цілочисельного типу, поле **tel** з номером телефону замовника – рядкового типу довжиною в 255 символів і поле **confirm**, яке містить інформацію про підтвердження замовлення – логічного типу.

Поля таблиці **deliveries**: **order_id** з номером замовлення – ключове, ненульове, унікальне поле цілочисельного типу з автоінкрементом, поле **field fio** з номером користувача, який здійснив замовлення, – ключове, ненульове цілочисельного типу, поле **address**, що зберігає адресу доставки товару, зазначену клієнтом, – рядкового типу довжиною 255 символів, поле **time**, що зберігає бажаний час доставки товару, – рядкового типу довжиною 255 символів, поле **date** з датою здійснення замовлення клієнтом – типу **DATE**, поле логічного типу **confirm**, що зберігає інформацію про доставку товару.

Таблиці **orders** і **deliveries** зведені на рис. 100.

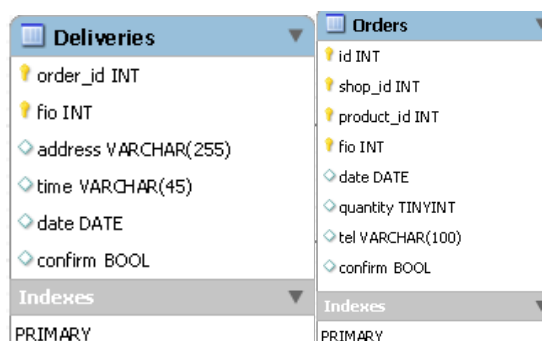


Рис. 100. Склад таблиць **orders** і **deliveries**

Зв'язки таблиць

Було створено базу даних, що складається із семи таблиць, тепер необхідно зв'язати таблиці, ми вже створили ключові поля цілочисельного типу, вони і стануть основою для зв'язування. Для того щоб зв'язати дві таблиці, наприклад, **products** і **product_type**, необхідно двічі клацнути лівою кнопкою миші на діаграму з таблицею **products** і вибрати вкладку **Foreign keys** (зовнішні ключі), далі в полі **Foreign key name** ввести унікальне ім'я зовнішнього ключа, двічі клацнути на вкладку **Referenced table** і вибрати таблицю **product_type**, потім у формі, що розміщена правіше, вибрати поле **type_id**, на яке посилається, і вибрати у впливаючому списку поле **id** (рис. 101).

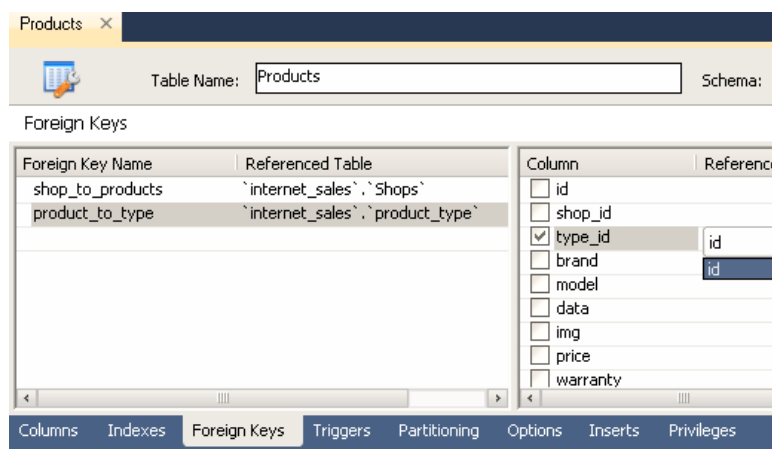


Рис. 101. Зв'язування таблиць **products** і **product_type** за допомогою **Foreign key**

Таким чином, обидва поля таблиці виявляються пов'язані, потім потрібно задати тип відносин зв'язку між таблицями, відкриємо вікно, клікнувши на зв'язок, що з'явився, між таблицями, і оберемо вкладку **Foreign Key**, і в розділі **Cardinality** оберемо тип зв'язку один до багатьох, і закриємо вікно. На діаграмі відобразиться зв'язок таблиць (рис. 102).

Аналогічним чином пов'яжемо всі ключові поля в таблицях, щоб вони були логічно взаємопов'язані, далі нам необхідно переконатися, що спроектована база даних відповідає третій нормальній формі.

Нормальна форма – властивість відношення в реляційній моделі даних, що характеризує його з погляду надмірності, яка потенційно може призвести до логічно помилкових результатів вибірки або зміни даних. Нормальна форма визначається як сукупність вимог, яким має задовольняти відношення.

У реляційній моделі відношення завжди перебуває в першій нормальній формі за визначенням поняття відношення. Що ж стосується різних таблиць, то вони можуть не бути правильними представленнями

відношень і, відповідно, можуть не перебувати в першій нормальній формі. Змінна відношення перебуває у другій нормальній формі тоді і тільки тоді, коли вона перебуває у першій нормальній формі і кожен не ключовий атрибут неприводимо (функціонально повно) залежить від її потенційного ключа. База даних перебуватиме в третій нормальній формі, якщо її приведено до другої нормальної форми, а кожен не ключовий стовпчик незалежний один від одного.

Таким чином наша база перебуває в третій нормальній формі, тому що кожен не ключовий стовпчик незалежний один від одного. Це наочно видно на діаграмі бази даних (рис. 103).

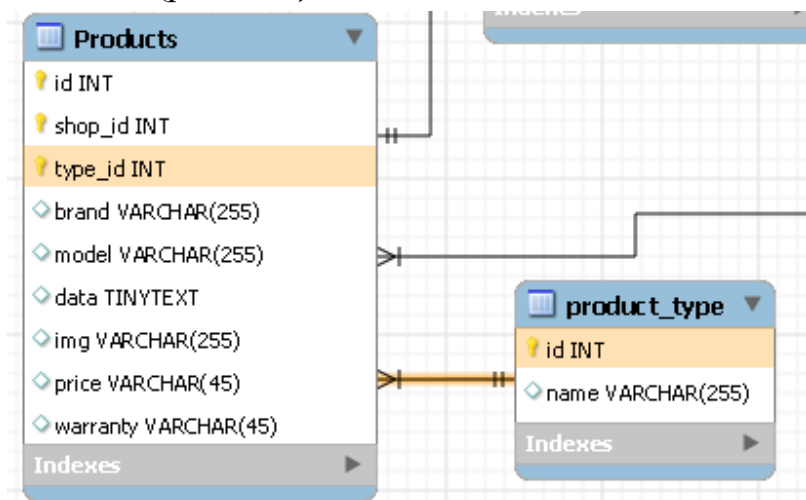


Рис. 102. Зв'язок таблиць

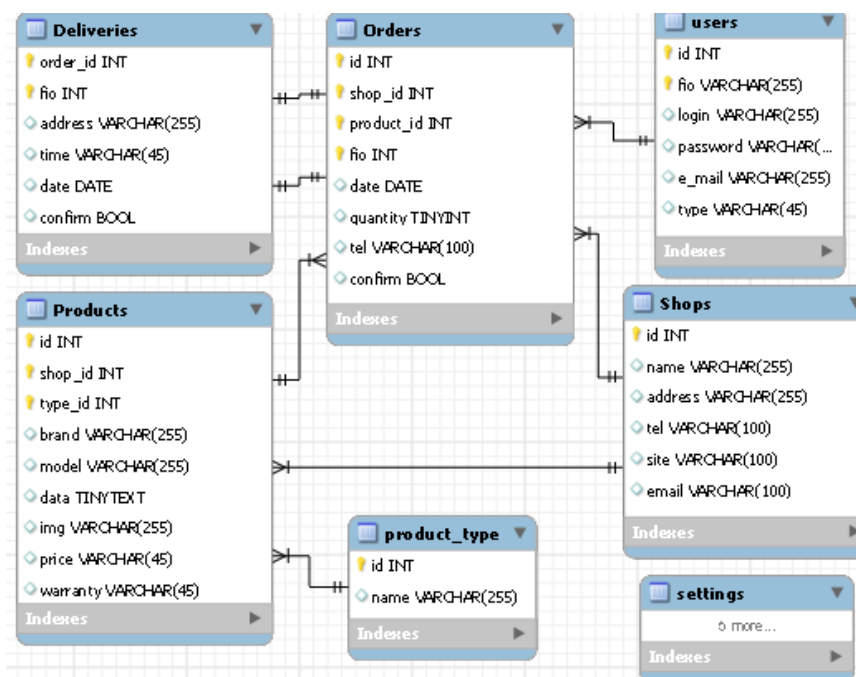


Рис. 103. Діаграма бази даних

Більшість таблиц перебувають у відношенні **один-до-багатьох**, за винятком таблиць **deliveries** і **orders**, що перебувають у відношенні **один-до-одного**, тому що доставленим може бути тільки одне замовлення, тобто в одного замовлення тільки одна доставка. Решта зв'язків наочно вказані вище.

Тепер завантажимо базу даних на сервер. Для цього створимо нове підключення до бази даних, клацнувши за посиланням **New connection** у стартовому вікні програми (рис. 104).



Рис. 104. Створення нового підключення до бази даних

Потім заповнимо поля у вікні, що відкрилося (рис. 105).

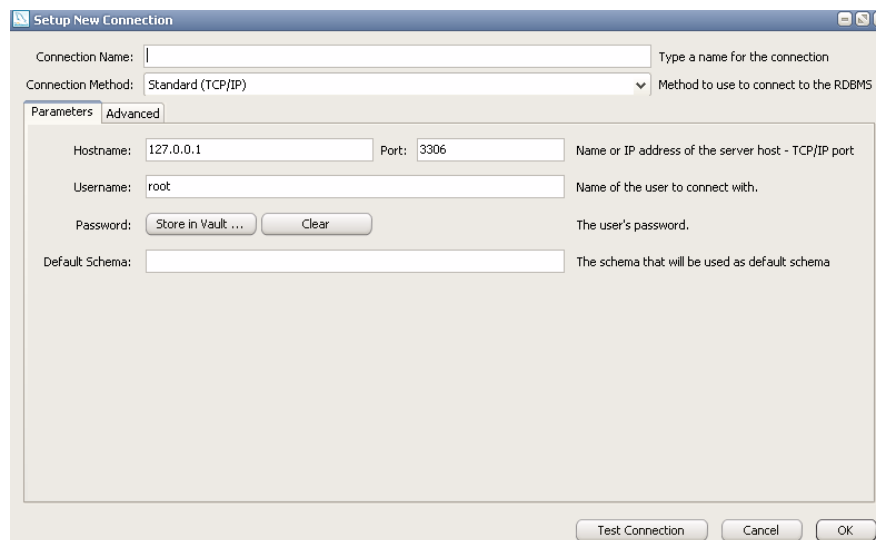


Рис. 105. Заповнення полів

Зазначимо ім'я з'єднання в полі **Connection Name**, виберемо метод з'єднання в переліку **Connection Method**, задамо ім'я хоста і порт у вкладці **Parameters**, вкажемо ім'я користувача і пароль, якщо він є, і натиснемо на

кнопку **OK**. Потім відкриємо вкладку **EER Diagram**, в панелі виберемо пункт **Database** і натиснемо на параметр **Forward Engineer** (рис. 106).

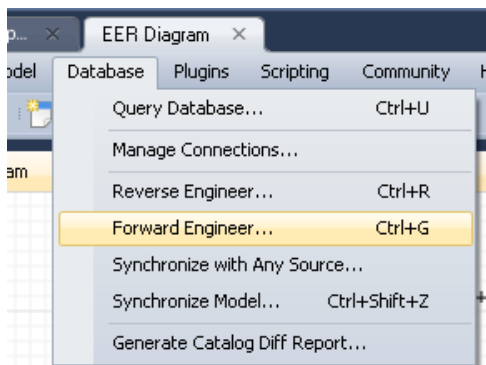


Рис. 106. Обрання параметрів **Forward Engineer**

Після того як з'явиться вікно, натискаємо на кнопку «**Next**», вибираємо параметр **Export MySQL Table Objects** і натискаємо на кнопку «**Next**» (рис. 107).

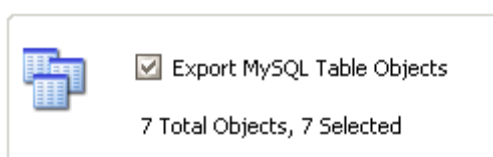


Рис. 107. Обрання параметру **Export MySQL Table Objects**

Після натискання кнопки з'явиться вкладка з SQL-кодом, можна зберегти його, натиснувши кнопку «**Save to file**», якщо це необхідно, а потім натиснути на кнопку «**Next**». З'явиться вікно з параметрами з'єднання (рис. 108).

Set Parameters for Connecting to a DBMS

Stored Connection: Select from saved connection settings

Connection Method: Method to use to connect to the RDBMS

Parameters ☒ Advanced

Hostname: Port: Name or IP address of the server host - TCP/IP port

Username: Name of the user to connect with.

Password: The user's password.

Рис. 108. Вікно з параметрами з'єднання

Перевіряємо, чи правильні параметри підключення і натискаємо на кнопку «**Execute**», якщо в SQL-кодi не міститься помилок, то після виконання коду ми побачимо вікно зі списком таблиць, інакше виведеться повідомлення про помилку. Тепер базу завантажено на сервер.

Виконання завдання 2

Створення та підключення бази даних за допомогою SQL SHELL (консолі)

1) Обрати PostgreSQL 16 – SQL Shell (psql) (рис. 109).

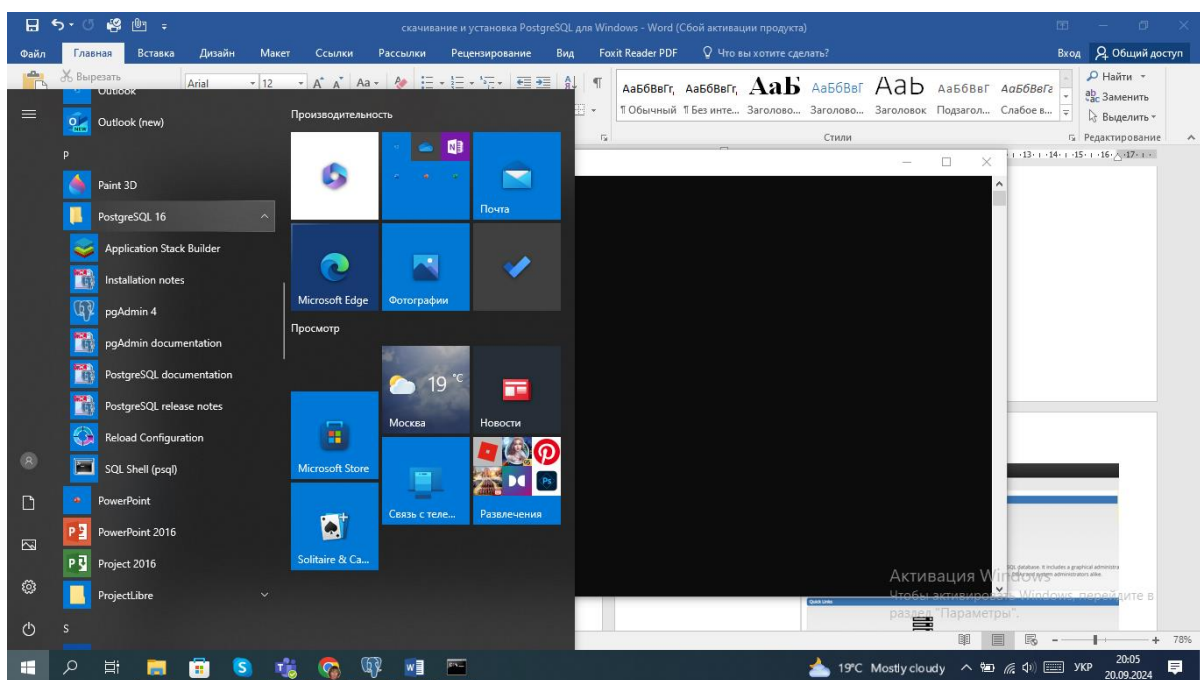


Рис. 109. Обрання режиму SQL Shell (psql)

2) При натисканні **Enter** використовуються установки, введені раніше (рис. 110).

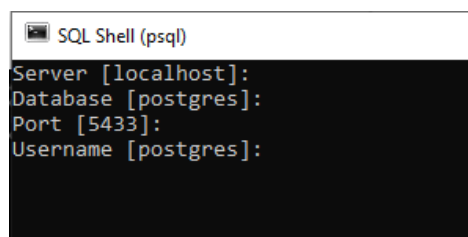


Рис. 110. Відображення установок, що були встановлені

3) Введення паролю, який створено попередньо (рис. 111).

```

SQL Shell (psql)
Server [localhost]:
Database [postgres]:
Port [5433]:
Username [postgres]:
Пароль пользователя postgres:
psql (16.3)
ПРЕДУПРЕЖДЕНИЕ: Кодовая страница консоли (866) отличается от основной
страницы Windows (1251).
8-битовые (русские) символы могут отображаться некорректно.
Подробнее об этом смотрите документацию psql, раздел
"Notes for Windows users".
Введите "help", чтобы получить справку.
postgres=#

```

Рис. 111. Введення паролю

4) При використанні `postgres=# \?` побачимо перелік команд, які використовуються (рис. 112).

```

Formatting
 \a                toggle between unaligned and aligned output mode
 \C [STRING]       set table title, or unset if none
 \f [STRING]       show or set field separator for unaligned query output
 \H               toggle HTML output mode (currently off)
 \set [NAME [VALUE]] set table output option
                  (border|columns|csv_fieldsep|expanded|fieldsep|
                  fieldsep_zero|footer|format|linestyle|null|
                  numericlocale|pager|pager_min_lines|recordsep|
                  recordsep_zero|tableattr|title|tuples_only|
                  unicode_border_linestyle|unicode_column_linestyle|
                  unicode_header_linestyle)
 \t [on|off]       show only rows (currently off)
 \T [STRING]       set HTML <table> tag attributes, or unset if none
 \x [on|off|auto] toggle expanded output (currently off)

Connection
 \c[connect] [(DATABASE|- USER|- HOST|- PORT|-)] | conninfo)
                  connect to new database (currently "postgres")
 \conninfo         display information about current connection
 \encoding [ENCODING] show or set client encoding
 \password [USERNAME] securely change the password for a user

Operating System
 \cd [DIR]         change the current working directory
 \setenv NAME [VALUE] set or unset environment variable
 \timing [on|off]   toggle timing of commands (currently off)
 \! [COMMAND]      execute command in shell or start interactive shell

Variables
 \prompt [TEXT] NAME prompt user to set internal variable
 \set [NAME [VALUE]] set internal variable, or list all if no parameters
 \unset NAME       unset (delete) internal variable

Large Objects
 \lo_export LOBOID FILE

```

Рис. 112. Перелік команд

5) Щоб переглянути які бази даних вже є, необхідно використовувати `postgres=# \l` (рис. 113).

```

postgres | postgres | UTF8 | libc | English_United States.1252 | English_United States.1252 |
template0 | postgres | UTF8 | libc | English_United States.1252 | English_United States.1252 |
          |          | =c/postgres | + | | |
          |          | postgres=CtC/postgres | | | |
template1 | postgres | UTF8 | libc | English_United States.1252 | English_United States.1252 |
          |          | =c/postgres | + | | |
          |          | postgres=CtC/postgres | | | |
testuser | testuser | UTF8 | libc | English_United States.1252 | English_United States.1252 |

```

Рис. 113. Перегляд переліку баз даних

6) Щоб створити нову базу даних, необхідно ввести:

```
postgres=# CREATE DATABASE avecoder;  
CREATE DATABASE  
postgres=#
```

7) Після введення `postgres=# \l` можна побачити, що створена база даних знаходиться в загальному переліку (рис. 114).

avecoder	postgres	UTF8	libc	English_United_States.1252	English_United_States.1252
postgres	postgres	UTF8	libc	English_United_States.1252	English_United_States.1252
template0	postgres	UTF8	libc	English_United_States.1252	English_United_States.1252
		=c/postgres	+		
		postgres=CtC/postgres			
template1	postgres	UTF8	libc	English_United_States.1252	English_United_States.1252
		=c/postgres	+		
		postgres=CtC/postgres			
testuser	testuser	UTF8	libc	English_United_States.1252	English_United_States.1252

Рис. 114. Перевірка наявності створеної бази даних в загальному переліку

8) Для під'єднання до цієї бази даних необхідно ввести:

`postgres=# \c avecoder` (рис. 115).

```
Вы подключены к базе данных "avecoder" как пользователь "postgres".  
avecoder=#  
  
avecoder=# \conninfo  
Вы подключены к базе данных "avecoder" как пользователь "postgres" (сервер "localhost": адрес ":::1", порт "5433").
```

Рис. 115. Підтвердження успішного під'єднання до бази даних

9) Для перевірки створеної бази даних зайдемо в pgAdmin (рис. 116).

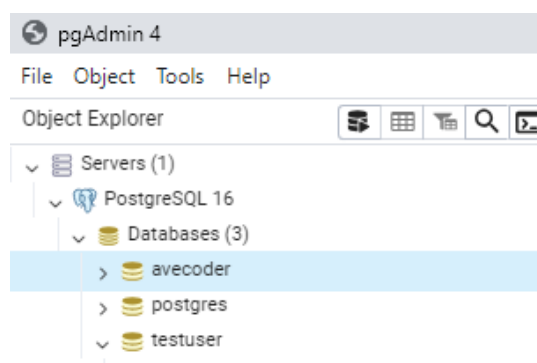


Рис. 116. Перевірка створеної бази даних через pgAdmin

Виконання завдання 3

Створення ролей та таблиць баз даних за допомогою pgAdmin

1) Заповнення вкладень

Помітити Login/Group Roles (18) та обрати
Create > Login/Group Role... (рис. 117 – рис. 120).

Group Role - Login/Group Roles

General Definition Privileges Membership Parameters Security SQL

Name avecoder

Comments

Рис. 117. Заповнення вкладки **General**

Group Role - Login/Group Roles

General Definition Privileges Membership Parameters Security SQL

Password

Account expires No Expiry

Please note that if you leave this field blank, then password will never expire.

Connection limit -1

Рис. 118. Заповнення вкладки **Definition**

Group Role - avecoder

General Definition Privileges Membership Parameters Security SQL

Can login? ☒

Superuser? ☒

Create roles? ☒

Create databases? ☒

Inherit rights from the parent roles? ☒

Can initiate streaming replication and backups? ☐

Bypass RLS? ☒

Рис. 119. Заповнення вкладки **Privileges**

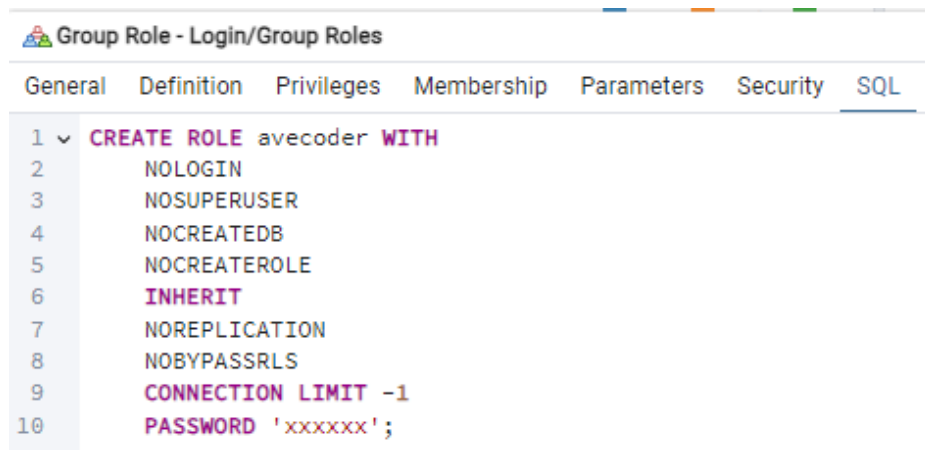


Рис. 120. Системне відображення результатів попередньо заповнених вкладок на вкладці **SQL**

Отримаємо наступну роль для бази даних avecoder –  avecoder .
Підключення до бази даних через тестюсер (рис. 121).

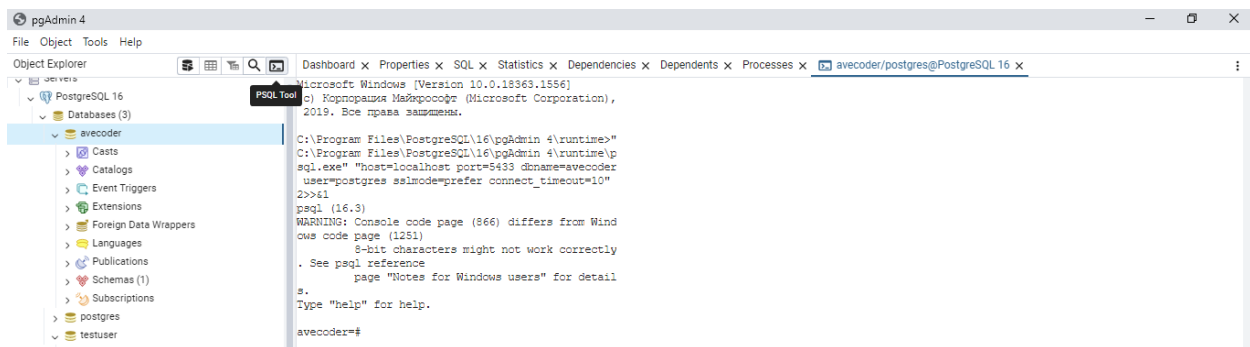


Рис. 121. Результат підключення бази даних через тестюсер

2) Створення таблиць (рис. 122).

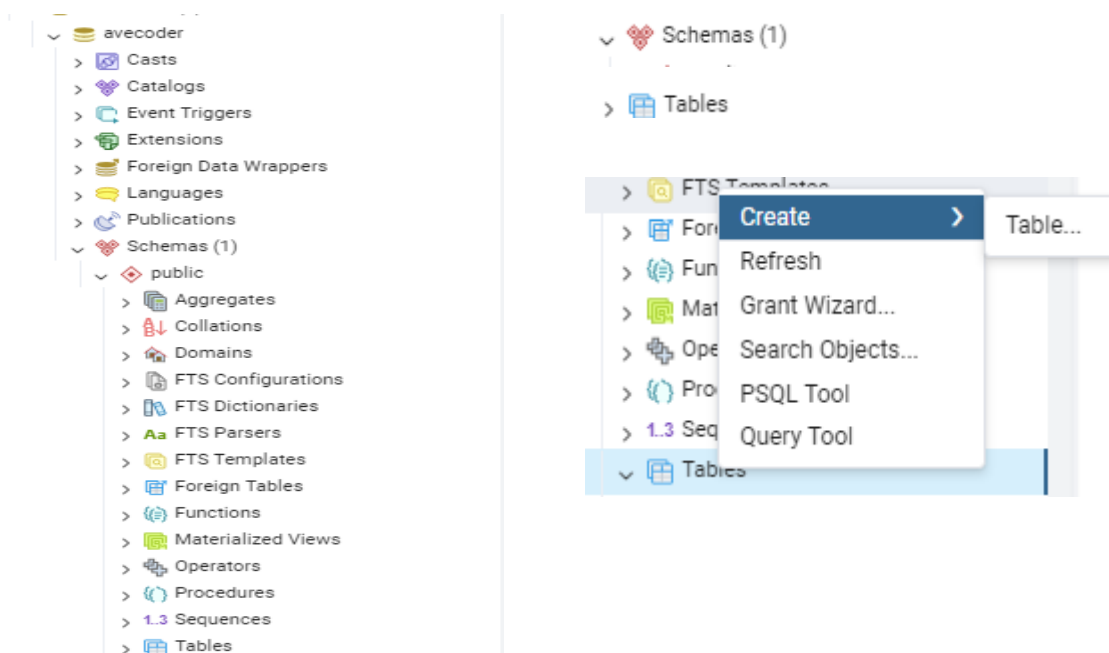


Рис. 122. Створення таблиці за допомогою команди **Create**

Створення таблиці «Види солодошів» згідно із табл. 3 та рис. 123.

Таблиця 3

sweets_types – види солодошів

Стовпець	Тип даних	Обов'язковість	Опис
id	integer	not null (повинно бути значення)	Ідентифікатор виду солодошів. PK
name	character varying	not null	Вид солодоші

Рис. 123. Заповнення вкладки **General** в таблиці **sweets_types**

Для стовпців **id** в таблиці встановлюємо обмеження (рис. 124).

Рис. 124. Заповнення вкладки **Columns** в таблиці **sweets_types**

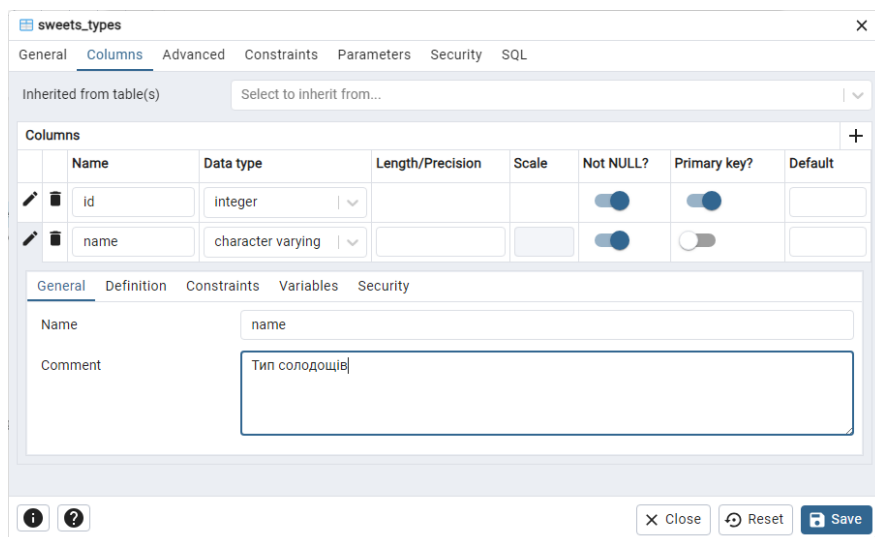


Рис. 124. Заповнення вкладки **Columns** в таблиці **sweets_types** (закінчення)

Створення таблиці «Виробники» згідно із табл. 4.

Таблиця 4

manufacturers – виробники

Стовпець	Тип даних	Обов'язковість	Опис
id	integer	not null	Ідентифікатор виробника. РК
name	character varying	not null	Виробник
phone	character varying		Телефон
adress	character varying		Адреса
city	character varying	not null	Населений пункт
country	character varying	not null	Країна

Для стовпців **id** в таблиці **manufacturers** встановлюємо обмеження (рис. 125).

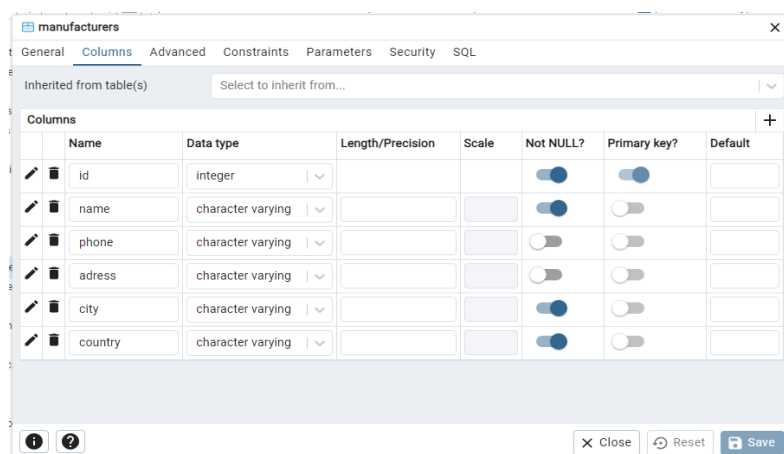


Рис. 125. Встановлення обмеження для стовпця **id** таблиці **manufacturers**

Створення таблиці «Склади» згідно із табл. 5.

Таблиця 5

storehouses – склади

Стовпець	Тип даних	Обов'язковість	Опис
id	integer	not null	Ідентифікатор складу. РК
name	character varying	not null	Назва складу
adress	character varying		Адреса
city	character varying	not null	Населений пункт
country	character varying	not null	Країна

Для стовпців **id** в таблиці **storehouses** встановлюємо обмеження (рис. 126).

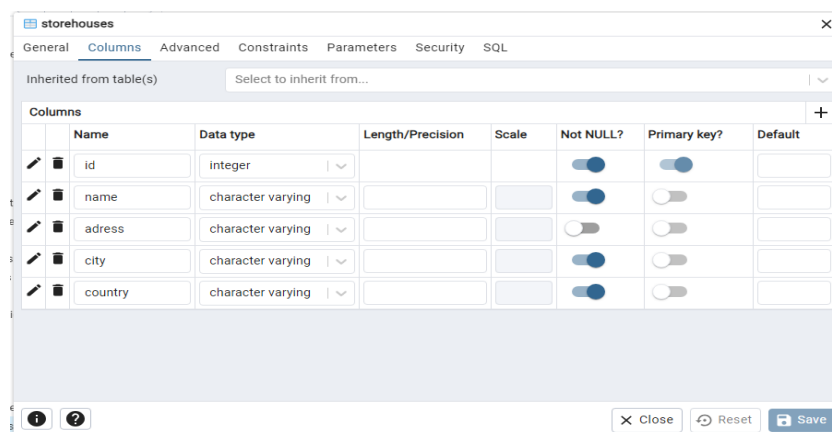


Рис. 126. Встановлення обмеження для стовпця **id** таблиці **storehouses**

Створення таблиці «Зв'язок виробника зі складом» згідно із табл. 6.

Таблиця 6

manufacturers_storehouses – зв'язок виробника зі складом

Стовпець	Тип даних	Обов'язковість	Опис
id	integer	not null	Ідентифікатор зв'язку. РК
storehouses_id	character varying	not null	Ідентифікатор складу. FK
manufacturers_id	character varying		Ідентифікатор виробника. FK

Для стовпців **id** в таблиці **manufacturers_storehouses** встановлюємо обмеження (рис. 127).

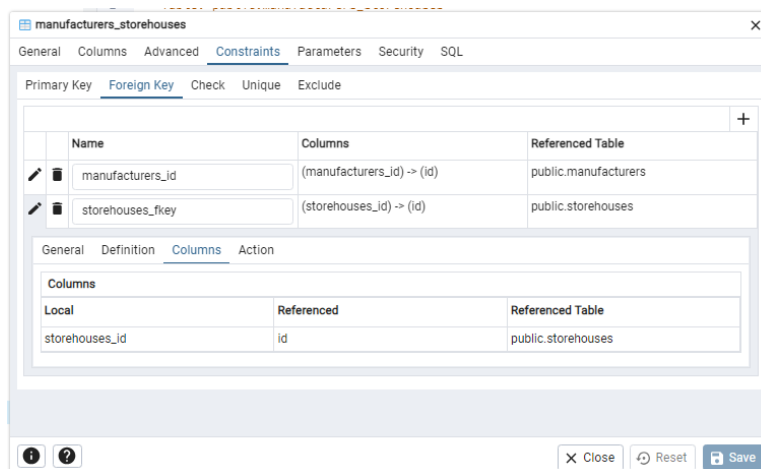


Рис. 127. Встановлення обмеження для стовпця **id** таблиці **manufacturers_storehouses**

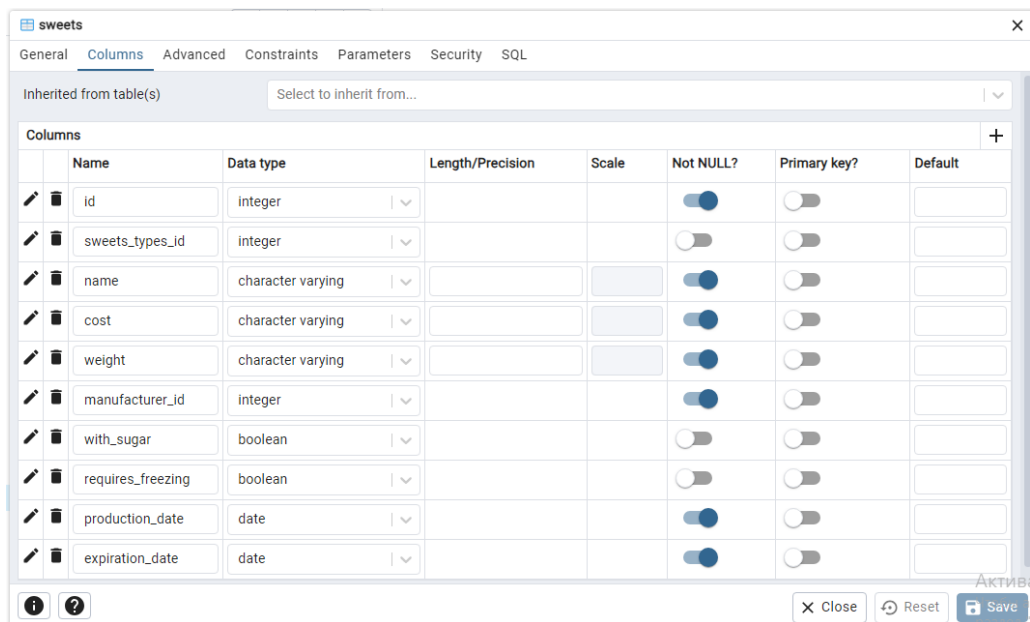
Створення таблиці «Солодощі» згідно із табл. 7.

Таблиця 7

sweets – солодощі

Стовпець	Тип даних	Обов'язковість	Опис
id	integer	not null	Ідентифікатор солодощі. РК
sweets_types_id	integer		Ідентифікатор виду. FK
name	character varying	not null	Назва солодощі
cost	character varying	not null	Вартість
weight	character varying	not null	Вага
manufacturers_id	integer	not null	Ідентифікатор виробника. FK
with_sugar	boolean		Із цукром? true – так, false – ні
requires_freezing	boolean		Потребує замороження? true – так, false – ні
production_date	date	not null	Дата виготовлення
expiration_date	date	not null	Термін придатності

Для стовпців **id** в таблиці **sweets** встановлюємо обмеження (рис. 128).



Name	Data type	Length/Precision	Scale	Not NULL?	Primary key?	Default
id	integer			☒	☒	
sweets_types_id	integer			☐	☐	
name	character varying			☒	☐	
cost	character varying			☒	☐	
weight	character varying			☒	☐	
manufacturer_id	integer			☒	☐	
with_sugar	boolean			☐	☐	
requires_freezing	boolean			☐	☐	
production_date	date			☒	☐	
expiration_date	date			☒	☐	

Рис. 128. Встановлення обмеження для стовпця **id** таблиці **sweets**

Контрольні запитання та завдання

1. Що собою являє поняття ключового поля?
2. Що собою являє первинний ключ?
3. Що собою являє зовнішній ключ?
4. В чому різниця між первинним та зовнішнім ключами?
5. Скільки в одній базі даних може бути первинних та зовнішніх ключів?
6. Як проводиться зв'язування між таблицями бази даних?
7. Надайте аналіз нормалізації представлених таблиць бази даних.

ЛАБОРАТОРНА РОБОТА № 4

ДОДАВАННЯ ДАНИХ У БАЗУ ДАНИХ ЗАСОБАМИ СУБД MYSQL ТА СУБД POSTGRESQL

Мета:

Придбати практичні навички із заповнення даними таблиці створеної бази даних засобами СУБД MySQL та СУБД PostgreSQL.

Завдання:

1. Додати дані в базу даних засобами СУБД MySQL
2. Додати дані в базу даних засобами СУБД PostgreSQL (pgAdmin).
3. Створити та видалити таблиці, заповнити таблиці даними за допомогою SQL Shell (консолі).

Виконання завдання 1

Додавання даних в базу даних засобами СУБД MySQL

Надалі опис особливостей діалекту SQL у СУБД MySQL буде вестися на прикладі навчальної бази даних. Такою базою даних обрано базу електронного магазину, що торгує комп'ютерними комплектуючими.

База даних комп'ютерного магазину складатиметься з чотирьох таблиць:

catalogs – список торговельних груп;

products – товарні позиції;

users – список зареєстрованих користувачів магазину;

orders – список здійснених угод.

Таблиця **catalogs** призначена для зберігання торгових груп, таких як «Материнські плати», «Процесори», «Відеокарти» тощо. Таблиця складається з двох полів:

id_catalog – унікальний номер;

name – ім'я роздела.

Оператор **CREATE TABLE**, що створює таблицю **catalogs**, наведено нижче. Обидва поля забезпечені атрибутом **NOT NULL**, оскільки невизначене значення для них неприпустиме.

```
CREATE TABLE catalogs (  
    id_catalog int(11) NOT NULL auto_increment,  
    `name` tinytext NOT NULL,
```

PRIMARY KEY (id_catalog),
) **DEFAULT CHARSET**=cp1251;

Таблиця **products** містить конкретні товарні позиції, такі як "Celeron 2.0GHz", "Intel Pentium 4 3.2GHz" і т. п. Таблиця складається із семи полів:

id_product – унікальний номер товарної позиції;

name – назва товарної позиції;

price – вартість;

count – кількість товарних позицій на складі;

mark – відносна оцінка товару;

description – опис;

id_catalog – номер торгової групи з таблиці **catalogs**, якій належить товарна позиція.

Оператор **CREATE TABLE**, що створює таблицю **products**, представлений нижче:

```
CREATE TABLE products (  
  id_product int(11) NOT NULL auto_increment,  
  `name` tinytext NOT NULL,  
  price decimal(7,2) default '0.00',  
  count int(11) default '0',  
  mark float(4,1) NOT NULL default '0.0',  
  description text,  
  id_catalog int(11) NOT NULL default '0',  
  PRIMARY KEY (id_product),  
  KEY id_catalog (id_catalog),  
  DEFAULT CHARSET=cp1251;
```

Ціна товарної позиції **price**, кількість товару на складі **count** і опис **description** забезпечені атрибутом **NULL**. Незважаючи на те, що в момент привезення часто буває невідома кількість товару на складі і його поточна ціна, відобразити факт наявності товару в прайс-листі необхідно. Так само й опис **description** може бути невідомим через те, що товар занадто новий і має суперечливий опис у джерелах.

Поле **id_catalog** встановлює зв'язок між таблицями **catalogs** і **products**. Знаючи унікальний номер каталогу **id_catalog**, завжди можна з'ясувати, які товарні позиції належать даній товарній групі і навпаки, до якої товарної групи належить товарна позиція з конкретним значенням **id_catalog**. MySQL дозволяє зробити це за допомогою нескладних SQL-запитів, які будуть розглянуті далі.

Таблиця **users** містить записи з інформацією про зареєстрованих покупців і складається з восьми полів:

id_user – унікальний номер покупця;

surname – прізвище покупця;

patronymic – по батькові покупця;

name – ім'я покупця;

phone – телефон покупця (якщо є);

email – e-mail покупця (якщо є);

url – домашня сторінка покупця (якщо є);

userstatus – статус покупця, поле типу ENUM, яке може приймати одне з чотирьох значень:

active – авторизований покупець, який може здійснювати покупки через Інтернет;

passive – неавторизований покупець (значення за замовчуванням), який здійснив процедуру реєстрації, але не підтвердив її і поки що не може здійснювати покупки в магазині через Інтернет, але йому доступні каталоги для перегляду;

lock – заблокований покупець, не може здійснювати покупки і переглядати каталоги магазину;

gold – активний покупець з гарною кредитною історією, якому надається знижка при наступних покупках у магазині.

Оператор **CREATE TABLE**, що створює таблицю **users**:

```
CREATE TABLE users (  
  id_user int(11) NOT NULL auto_increment,  
  surname tinytext,  
  patronymic tinytext,  
  `name` tinytext,  
  phone varchar(12) default NULL,  
  email tinytext,  
  url tinytext,  
  `status` enum('active','passive','lock','gold') default 'active',  
  PRIMARY KEY (id_user)  
 DEFAULT CHARSET=cp1251;
```

Поля **phone**, **email** і **url** забезпечені атрибутом **NULL**, оскільки відвідувач має право не вказувати цю інформацію. Якщо ця інформація не вказана, то вона має невизначений характер: телефону, e-mail і домашньої сторінки в покупця може просто не бути, з іншого боку, він міг не захотіти їх

вказувати. Тому, щоб вказати невизначеність цієї інформації, у поле слід помістити значення **NULL**. Усі інші поля отримують атрибут **NOT NULL**, навіть якщо він не вказується.

Таблиця **orders** містить інформацію про покупки, здійснені в магазині, і включає п'ять полів:

id_order – унікальний номер сделки;

id_user – номер пользователя из таблицы users;

ordertime – время совершения сделки;

number – число приобретенных товаров;

id_product – номер товарной позиции из таблицы products.

Оператор **CREATE TABLE**, создающий таблицу **orders**:

```
CREATE TABLE orders (  
    id_order int(11) NOT NULL auto_increment,  
    id_user int(11) NOT NULL default '0',  
    ordertime datetime NOT NULL default '0000-00-00 00:00:00',  
    number int(11) NOT NULL default '0',  
    id_product int(11) NOT NULL default '0',  
    PRIMARY KEY (id_order),  
    KEY id_product (id_product),  
    KEY id_user (id_user)  
    DEFAULT CHARSET=cp1251;
```

Усі стовпці таблиці **orders** забезпечені атрибутом **NOT NULL**, тому що під час здійснення купівлі вся інформація має бути доступною і занесена в таблицю.

У таблиці **orders** встановлюється зв'язок одразу з двома таблицями: **users** за рахунок поля **id_user**, і **products** за рахунок поля **id_product**. Це дасть змогу для кожної покупки відновити покупця і придбаний товар. З іншого боку, можна дізнатися кількість покупок, здійснених кожним із покупців, а також частоту купівлі тієї чи іншої товарної позиції, що дасть змогу збільшити або зменшити закупівлю товару і в такий спосіб провести оптимізацію роботи магазину.

Однорядковий оператор INSERT

Після успішного створення бази даних і таблиць перед розробником постає завдання заповнення таблиць даними. Традиційно в реляційних базах даних для здійснення цієї операції застосовують три підходи:

– однорядковий оператор **INSERT** додає в таблицю даних новий запис;

– багаторядковий оператор **INSERT** додає в таблицю даних кілька нових записів;

– пакетне завантаження даних слугує для додавання в таблицю даних із зовнішнього файлу.

Однорядковий оператор **INSERT** може використовуватися в кількох формах. Спрощений синтаксис першої форми має такий вигляд:

```
INSERT [INTO] tbl [(col_name, ...)] VALUES (expression,...)
```

Цей оператор вставляє новий запис у таблицю **tbl**, значення запису перераховуються в списку **expression**, порядок проходження стовпців може задаватися списком **col_name**. Значення передаються в список після ключового слова **VALUES**.

Розглянемо процес вставки на прикладі таблиці **catalogs**, яка має два поля:

id_catalog – первинний ключ таблиці **catalogs**, забезпечений атрибутом **AUTO_INCREMENT**, значення за замовчуванням – **NULL**;

NAME – назва каталогу, значення за замовчуванням – порожній рядок.

Перегляд даних, поміщених у таблицю **catalogs**, здійснюється за допомогою запиту **SELECT * FROM catalogs**.

Додати новий запис у таблицю **catalogs** можна за допомогою такого запиту:

```
INSERT INTO catalogs VALUES (1, 'Процесори');
```

```
SELECT * FROM catalogs;
```

```
+-----+-----+
| id_catalog | name      |
+-----+-----+
|          1 | Процесори |
+-----+-----+
```

У таблицю **catalogs** додався новий запис із первинним ключем **id_catalogs**, що дорівнює одиниці, і значенням поля **name** – «Процесори». Рядкові значення необхідно поміщати в лапки, тоді як числові значення допускається використовувати без них.

Список стовпців **col_name**, розміщений після імені таблиці, дає змогу змінити порядок проходження стовпців під час додавання:

```
INSERT INTO catalogs (name, id_catalog)
```

```
VALUES ('Материнські плати', 2);
```

```
SELECT * FROM catalogs;
```

```
+-----+-----+
```

id_catalog	name
1	Процесори
2	Материнські плати

Як видно, порядок проходження стовпців було змінено, спочатку було додано назву розділу `name` і лише потім первинний ключ таблиці **id_catalog**. Слід пам'ятати, що первинний ключ таблиці є унікальним значенням, і додавання вже існуючого значення призведе до помилки:

ERROR 1062 (23000): Duplicate entry '2' for key 1

Під час додавання нового запису з унікальними індексами вибір такого унікального значення може бути непростим завданням. Для того щоб не здійснювати додатковий запит, спрямований на виявлення максимального значення первинного ключа для генерації нового унікального значення, в MySQL введено механізм його автоматичної генерації. Для цього достатньо забезпечити первинний ключ **id_catalog** атрибутом **AUTO_INCREMENT**. Тоді при створенні нового запису як значення **id_catalog** достатньо передати **NULL** або 0 – поле автоматично отримає значення, що дорівнює максимальному значенню стовпця **id_catalog**, плюс одиниця:

INSERT INTO catalogs VALUES (NULL, 'Відеоадаптери');

SELECT * FROM catalogs;

id_catalog	name
1	Процесори
2	Материнські плати
3	Відеоадаптери

При вставці оператором **INSERT** не обов'язково вказувати всі стовпці. Ті стовпці, які не вказані в списку, отримають значення за замовчуванням. Так, тут значення **id_catalog** не передається оператору **INSERT**:

INSERT INTO catalogs (name) VALUES ('Жорсткі диски');

SELECT * FROM catalogs;

id_catalog	name
1	Процесори

2	Материнські плати
3	Відеоадаптери
4	Жорсткі диски

Так як **id_catalog** за замовчуванням набуває значення **NULL**, яке викликає механізм **AUTO_INCREMENT**, то поле **id_catalog** автоматично отримує значення 4 (максимальне значення 3 плюс 1). Допускається додавання повністю порожніх рядків:

```
INSERT INTO catalogs () VALUES ();
SELECT * FROM catalogs;
```

id_catalog	name
1	Процесори

5	
---	--

Оскільки текстове поле **name** за замовчуванням приймає порожній рядок, новий запис містить порожній запис. Значення за замовчуванням можна підставити, якщо передати як значення ключове слово **DEFAULT**:

```
INSERT catalogs (id_catalog, name) VALUES (DEFAULT, DEFAULT);
SELECT * FROM catalogs;
```

id_catalog	name
1	Процесори

6	
---	--

При виборі стовпця, забезпеченого атрибутом **AUTO_INCREMENT**, у програмістів-початківців часто виникає питання, що відбувається, коли досягається межа типу. У цьому випадку виникає помилка під час додавання нового запису в таблицю, оскільки унікальних значень у межах типу не залишається. На практиці досягти цієї межі можна тільки для дуже коротких типів, наприклад, **TINYINT**. Вибір типу **INT** і **BIGINT** навіть у разі щоденного додавання в таблицю 10 000 нових записів призведе до вичерпання унікальних значень тільки через 588 і 2 500 000 000 000 000

років, відповідно. При цьому навіть якщо таблиця міститиме тільки один стовпець, займаний об'єм даних досягне 8 Гбайт для типу **INT** та 67 108 864 Тбайт для типу **BIGINT**.

Крім констант, у списку значень **expression** можуть виступати вирази за участю стовпців зі списку **col_name**. Приклад наведено нижче:

```
INSERT catalogs (id_catalog, name) VALUES (7, id_catalog*3);
```

```
INSERT catalogs (id_catalog, name) VALUES (name*3, 7);
```

```
SELECT * FROM catalogs;
```

```
+-----+-----+
| id_catalog | name      |
+-----+-----+
|      1 | Процесори |
...
|      7 | 21        |
|      8 | 7         |
+-----+-----+
```

Так, для першого запису поле **id_catalog** набуває значення 6, а поле **name** значення 21 (7х3). Посилатися у виразі можна тільки на стовпці, розташовані лівіше, посилання на праві стовпці дає 0 (у разі стовпця **id_catalog** це призвело до автоматичної генерації нового унікального значення 8 замість очікуваних 21).

Якщо під час вставки нового значення задіяно механізм **AUTO_INCREMENT**, то скористатися новозгенерованим значенням не вдасться. Так спочатку полю **id_catalog** присвоюється значення 0, і лише після того, як запис додано – генерується нове значення:

```
INSERT catalogs (id_catalog, name) VALUES (NULL, id_catalog*3);
```

```
SELECT * FROM catalogs;
```

```
+-----+-----+
| id_catalog | name      |
+-----+-----+
|      1 | Процесори |
|      9 | 0         |
+-----+-----+
```

Ще однією формою оператора **INSERT** є синтаксис із ключовим словом **SET**:

```
INSERT [INTO] tbl SET col_name=expressin
```

Оператор заносить у таблицю **tbl** новий запис, стовпчик **col_name** у якому отримує значення **expression**.

Наступні два запити виконують одне й те саме:

```
INSERT INTO catalogs SET id_catalog = 18, name = 'Мережеві карти'
```

```
INSERT INTO catalogs VALUES (18, 'Мережеві карти');
```

Форма оператора **INSERT** із ключовим словом **SET** дає змогу гнучкіше додавати записи. Якщо ім'я стовпця після **SET** не зустрічається, полю присвоюється значення за замовчуванням.

Багаторядковий оператор INSERT

Багаторядковий оператор **INSERT** збігається за формою з однорядковим оператором. У ньому використовується ключове слово **VALUES**, після якого додається не один, а кілька списків **expression**. Так, у наступному лістингу додається відразу п'ять записів за допомогою одного оператора **INSERT**:

```
INSERT INTO catalogs VALUES (0, 'Процесори'),
```

```
-> (0, 'Материнські плати'),
```

```
-> (0, 'Відеоадаптери'),
```

```
-> (0, 'Жорсткі диски'),
```

```
-> (0, 'Оперативна пам'ять');
```

Точно так само, як і в разі однорядкового варіанта, допускається змінювати порядок і склад списку доданих:

```
INSERT INTO catalogs (name) VALUES ('Процесорb'),
```

```
-> ('Материнські плати'),
```

```
-> ('Відеоадаптери'),
```

```
-> ('Жорсткі диски'),
```

```
-> ('Оперативна пам'ять');
```

Виконання завдання 2

Додати дані в базу даних засобами СУБД PostgreSQL (pgAdmin)

Наповнюємо таблиці даними за допомогою SQL-скрипта. Використовуємо оператор **INSERT INTO ... VALUES**.

У pgAdmin заходимо в таблицю **sweets_types** і створюємо скрипт Insert: Схеми -> public -> Таблиці -> sweets_types (права кнопка миші) -> Scripts -> Insert.

Аналогічно наповнюємо даними інші таблиці.

Нижче наведені рисунки як це зробити в pgAdmin (рис. 129).

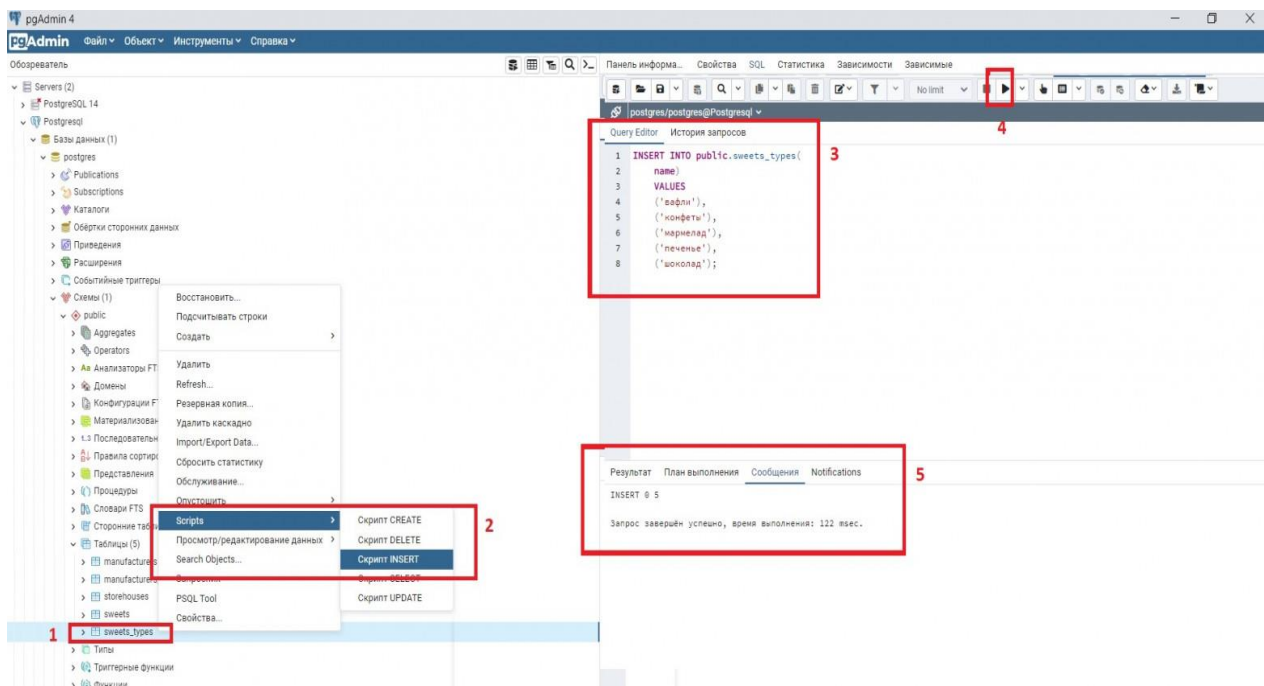


Рис. 129. Наповнення даними таблиці

Створення скриптів SQL на наповнення даними таблиць

1) Заповнення таблиці **sweets_types** (рис. 130).

```
INSERT INTO public.sweets_types(
    name)
VALUES
('вафлі'),
('цукерки'),
('мармелад'),
('печиво'),
('шоколад');
```

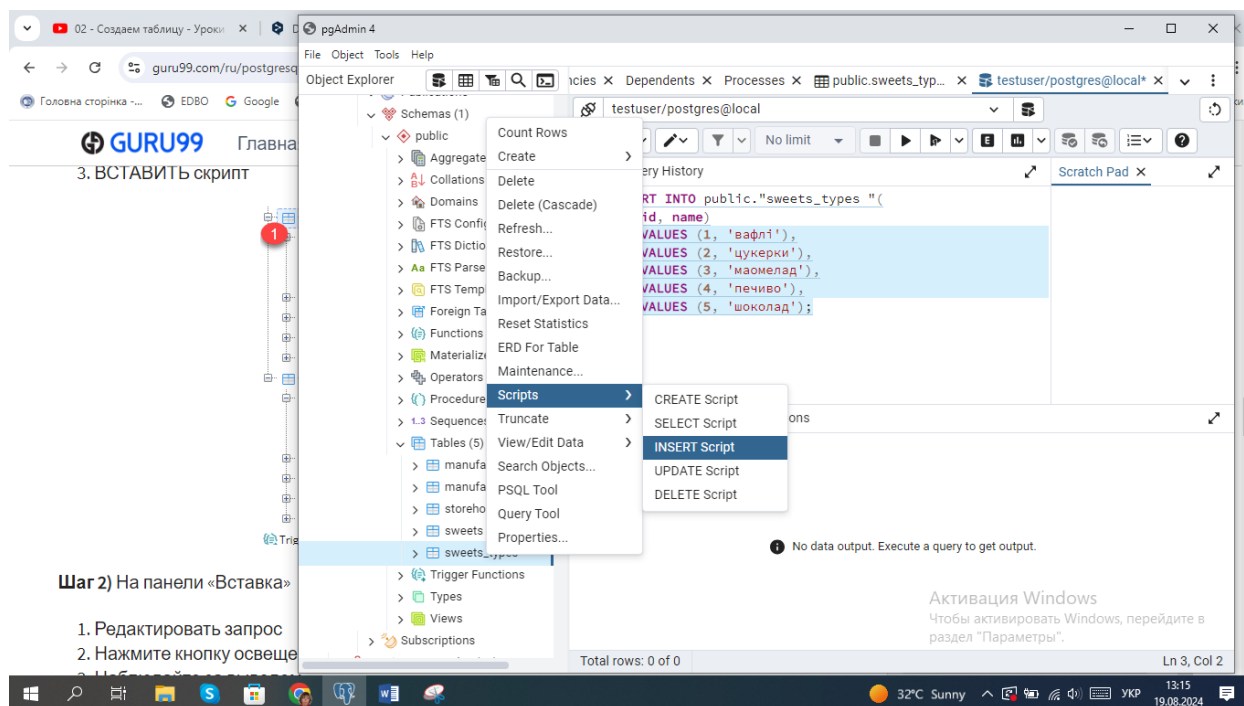


Рис. 130. Заполнения таблиці **sweets_types**

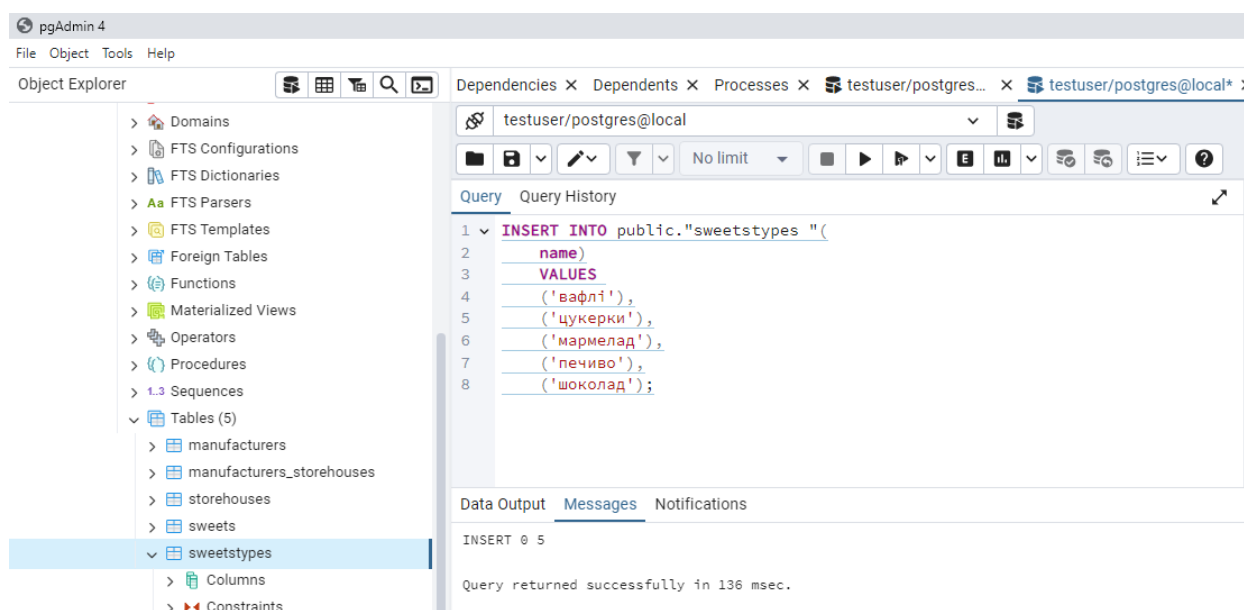


Рис. 130. Заполнения таблиці **sweets_types** (продовження)

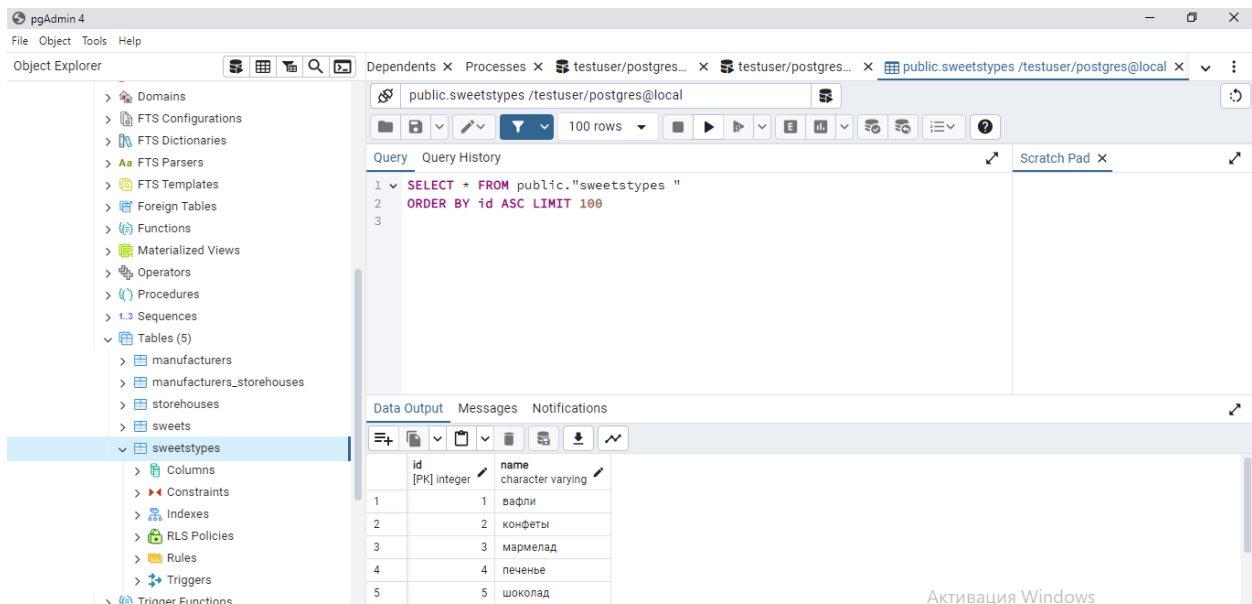


Рис. 130. Заповнення таблиці **sweets_types** (закінчення)

2) Заповнення таблиці **storehouses** (рис. 131).

INSERT INTO public.storehouses(

name, adress, city, country)

VALUES

('ХК-1', 'вул. Севастопольська, б. 134', 'м. Харків', 'Україна'),

('КК-2', 'вул. Гузара, б. 23', 'м. Київ', 'Україна'),

('ПК-1', 'вул. Ясенєва, б. 5', 'м. Полтава', 'Україна'),

('ОК-1', 'вул. 13 Фонтану, б. 67', 'м. Одеса', 'Україна');

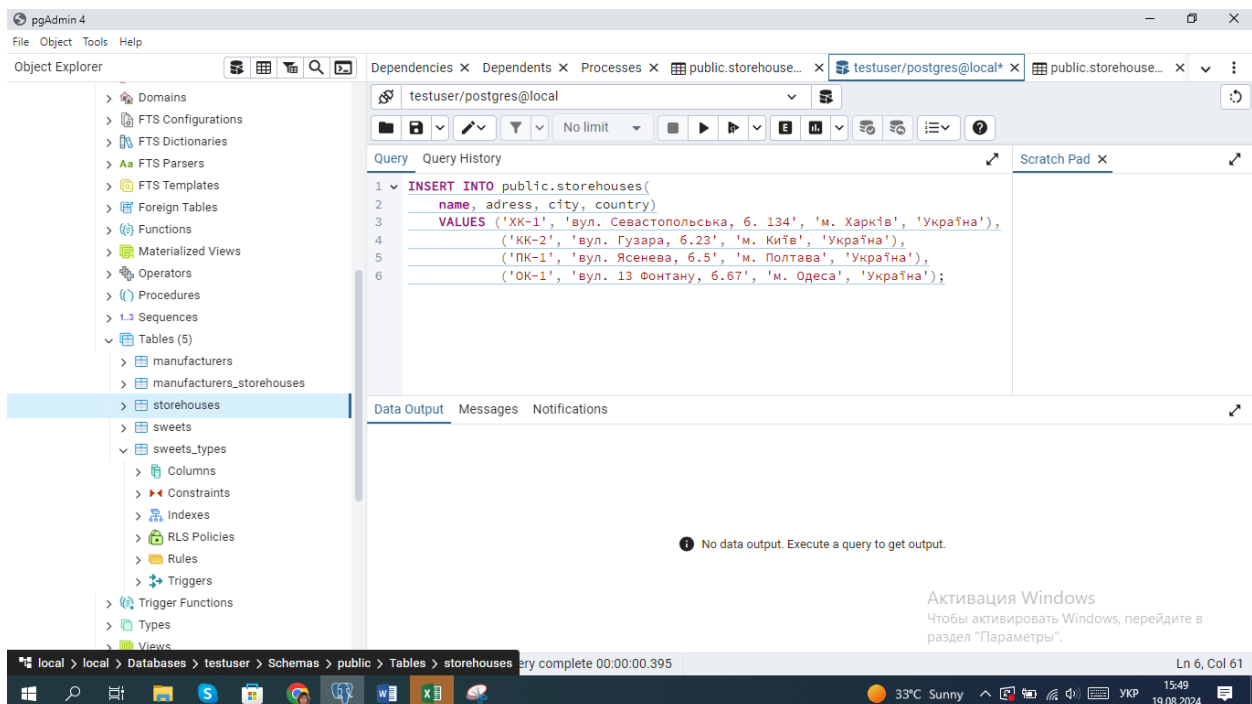


Рис. 131. Заповнення таблиці **storehouses**

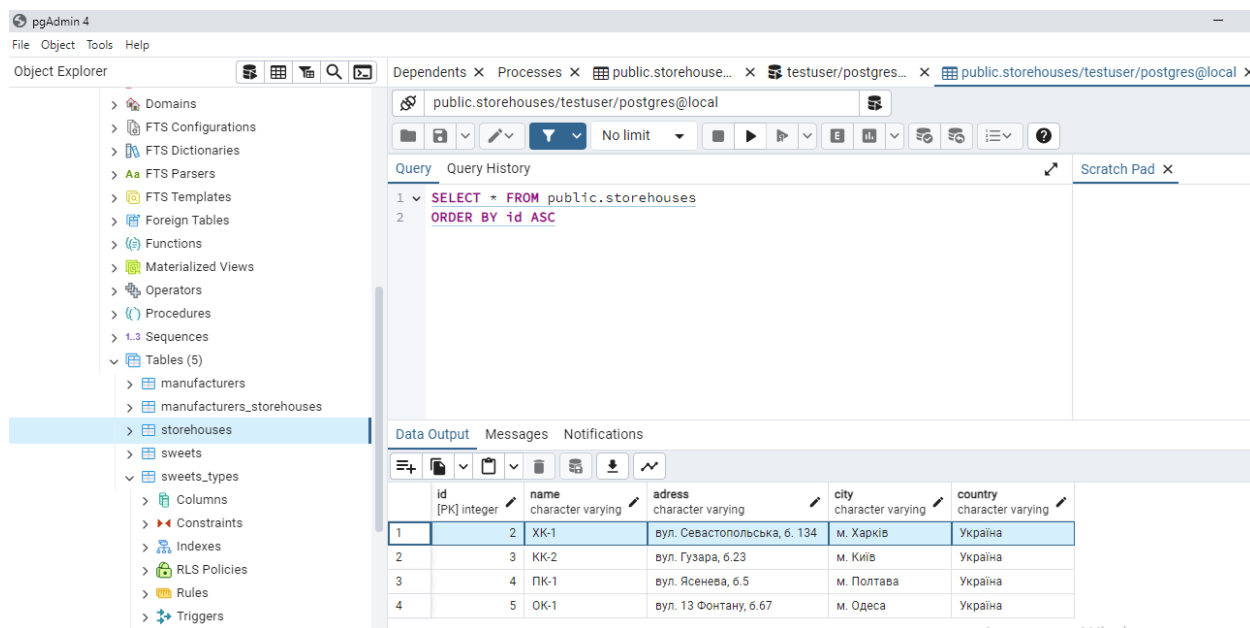


Рис. 131. Заповнення таблиці **storehouses** (закінчення)

3) Заповнення таблиці **manufacturers** (рис. 132).

INSERT INTO public.manufacturers(

name, phone, address, city, country)

VALUES

('Ведмедик', '+380577812390', '61001, майдан Конституції, 6.2', 'м. Харків', 'Україна'),

('Ласуня', '+38044456899', '24105, Борщагівська вулиця, б. 75', 'м. Київ', 'Україна'),

('Смаколик', '+380588905634', '62013, Шефська вулиця, б. 5А', 'м. Полтава', 'Україна'),

('Білочка', '+380566783412', '10923, Проектна вулиця, б.15', 'м. Одеса', 'Україна');

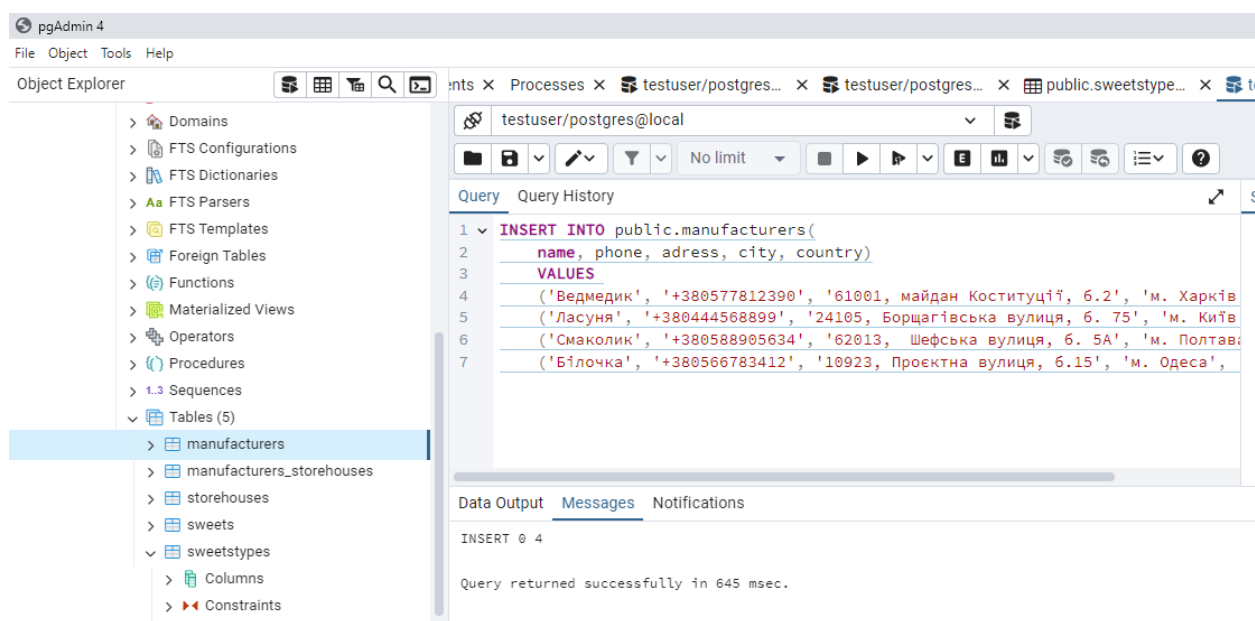


Рис. 132. Заповнення таблиці **manufacturers**

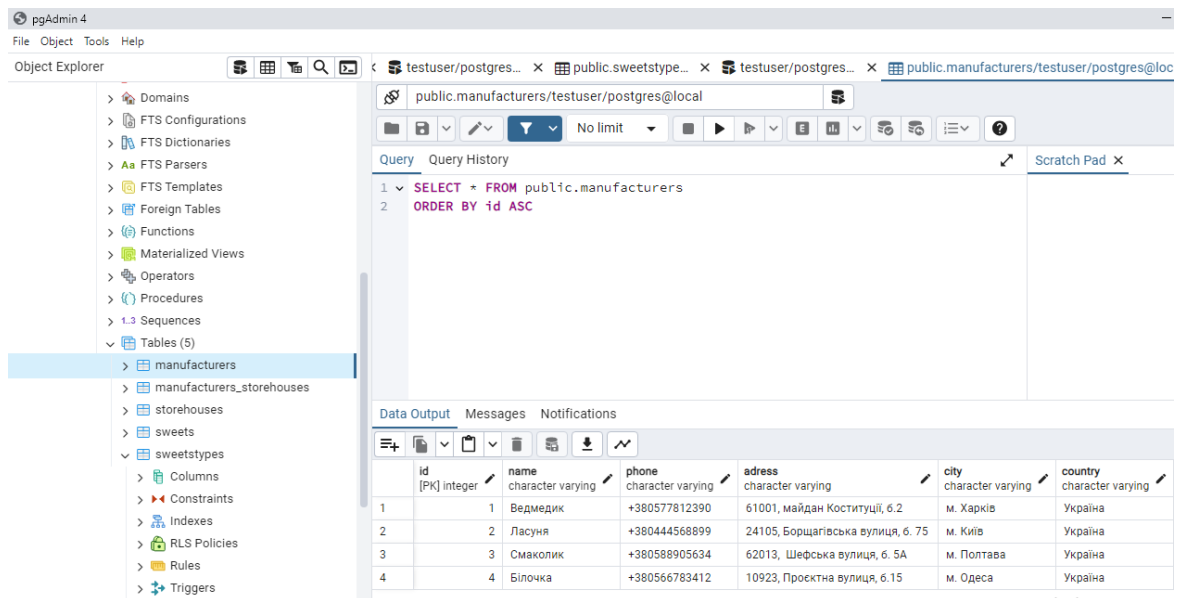


Рис. 132. Заповнення таблиці **manufacturers** (закінчення)

4) Заповнення таблиці **manufacturers_storehouses** (рис. 133).

```
INSERT INTO public.manufacturers_storehouses(
    storehouses_id, manufacturers_id)
VALUES
(1, 1),
(2, 2),
(3, 3),
(1, 2),
(2, 1);
```

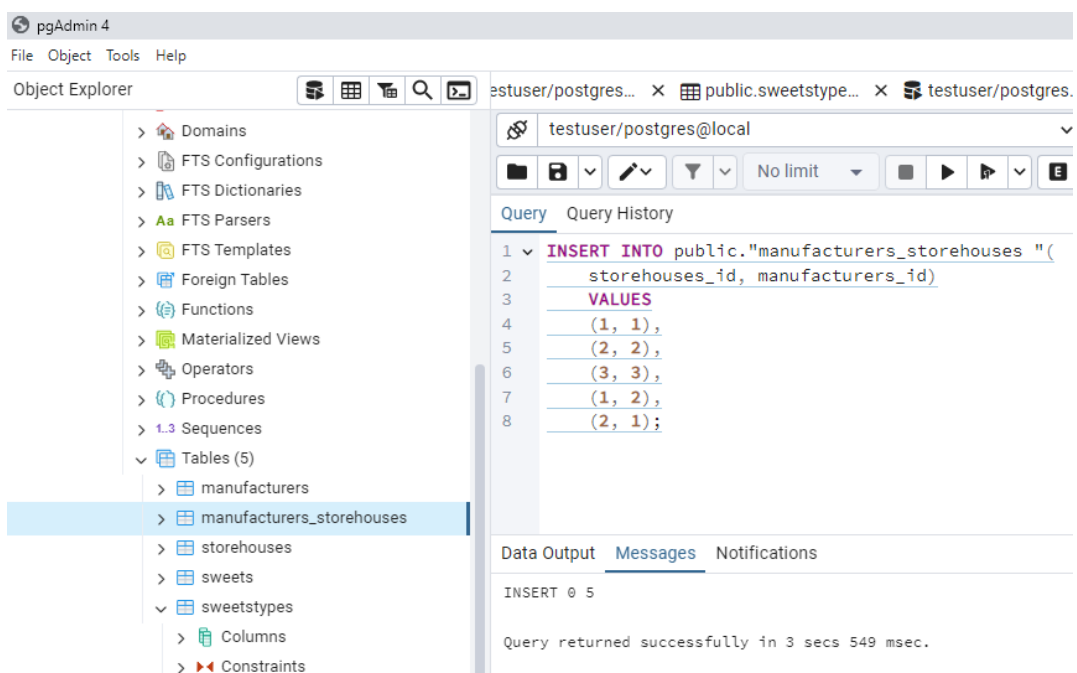


Рис. 133. Заповнення таблиці **manufacturers_storehouses**

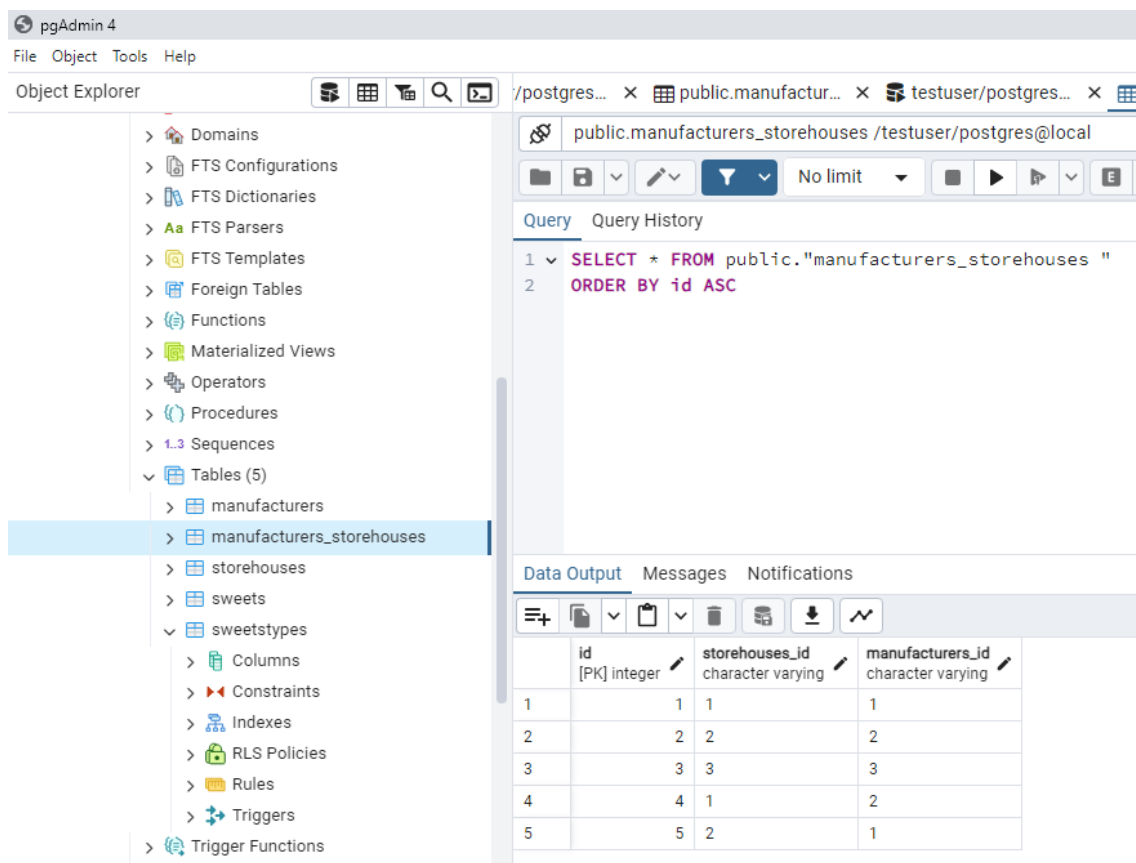


Рис. 133. Заповнення таблиці **manufacturers_storehouses** (закінчення)

5) Заповнення таблиці **sweets** (рис. 134).

```
INSERT INTO public.sweets(
    sweets_types_id, name, cost, weight, manufacturer_id, with_sugar, requires_freezing,
    production_date, expiration_date)
```

VALUES

```
(1, 'Артек', '30', '1кг', 1, false, false, '2024-09-03', '2025-03-15'),
(2, 'Корівка', '75', '1кг', 1, true, true, '2024-09-03', '2025-03-03'),
(3, 'Хелейні', '56', '1кг', 1, true, false, '2024-09-03', '2025-03-14'),
(4, 'Вівсяне', '120', '200', 1, false, true, '2024-09-03', '2022-04-04'),
(5, 'Millennium (фундук молочні)', '145', '570', 1, true, false, '2021-03-03', '2021-12-03'),
(1, 'Шоколадні', '115', '200', 2, false, false, '2024-09-03', '2022-05-15'),
(2, 'Південна ніч', '155', '300', 2, true, true, '2024-09-03', '2022-05-03'),
(3, 'Лимонні дольки', '117', '500', 2, true, false, '2024-09-03', '2022-04-14'),
(4, 'Танучий сніг', '129', '250', 2, false, true, '2024-09-03', '2022-04-04'),
(5, 'Millennium (фундук чорний)', '77', '1плитка', 2, true, false, '2021-02-03', '2021-12-03'),
(1, 'Наполітанки', '60', '1кг', 3, false, false, '2024-09-03', '2022-05-15'),
(2, 'Червоний мак', '190', '1кг', 3, true, true, '2024-09-03', '2025-03-03');
```

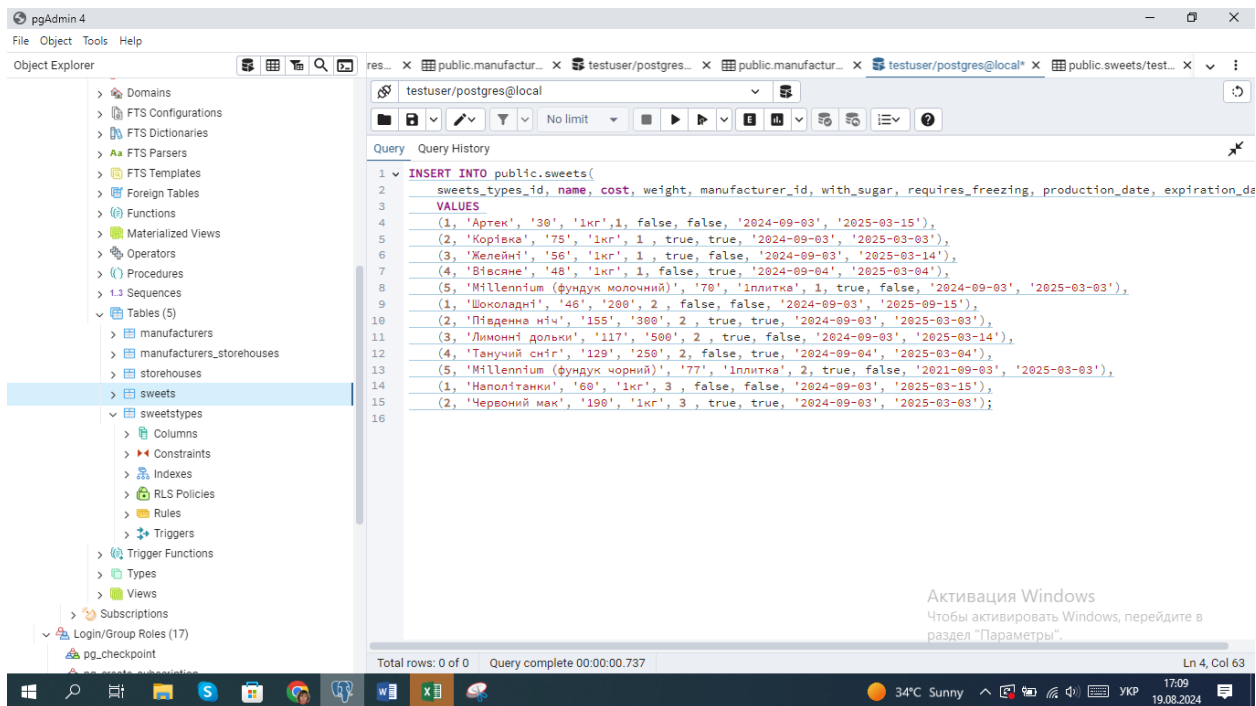


Рис. 134. Заповнення таблиці **sweets**

Query:

```

1 SELECT * FROM public.sweets
2 ORDER BY id ASC

```

id	sweets_types_id	name	cost	weight	manufacturer_id	with_sugar	requires_freezing	production_date	expiration_date
1	1	Арте́к	30	1kr	1	false	false	2024-09-03	2025-03-15
2	2	Корі́вка	75	1kr	1	true	true	2024-09-03	2025-03-03
3	3	Желе́йні	56	1kr	1	true	false	2024-09-03	2025-03-14
4	4	Ві́скане	48	1kr	1	false	true	2024-09-04	2025-03-04
5	5	Мі́лленіум (фундук мо...	70	1плитка	1	true	false	2024-09-03	2025-03-03
6	6	Шоколадні́	46	200	2	false	false	2024-09-03	2025-09-15
7	7	Пі́вденна ні́ч	155	300	2	true	true	2024-09-03	2025-03-03
8	8	Лимонні́ дольки	117	500	2	false	false	2024-09-03	2025-03-14
9	9	Тану́чий сні́г	129	250	2	false	true	2024-09-04	2025-03-04
10	10	Мі́лленіум (фундук чо...	77	1плитка	2	true	false	2021-09-03	2025-03-03
11	11	Напо́літанські	60	1kr	3	false	false	2024-09-03	2025-03-15
12	12	Черво́ний ма́к	190	1kr	3	true	true	2024-09-03	2025-03-03

Рис. 134. Заповнення таблиці **sweets** (закінчення)

✓ Таблиці наповнено.

Виконання завдання 3

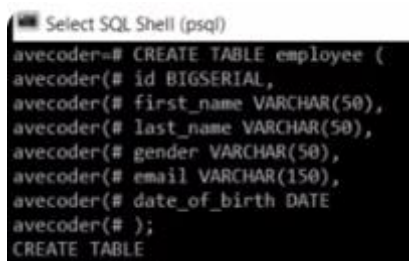
Створення та видалення таблиць, заповнення таблиць даними за допомогою SQL Shell (консолі)

1. Створення таблиці **Співробітники (employee)**

У якості **id** застосуємо змінну **BIGSERIAL** (ідентифікатор, що самозбільшується).

Поля таблиці **first_name**, **last_name**, **gender** і **email** з типом **VARCHAR** – складається з букв і цифр.

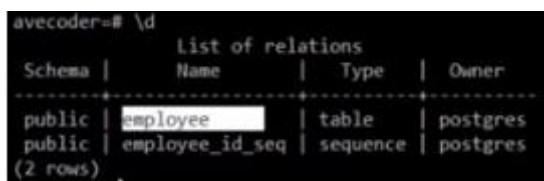
Далі йде повідомлення про успішне створення таблиці із зазначеними полями (**CREATE TABLE**) (рис. 135).



```
Select SQL Shell (psql)
avecoder=# CREATE TABLE employee (
avecoder(# id BIGSERIAL,
avecoder(# first_name VARCHAR(50),
avecoder(# last_name VARCHAR(50),
avecoder(# gender VARCHAR(50),
avecoder(# email VARCHAR(150),
avecoder(# date_of_birth DATE
avecoder(# );
CREATE TABLE
```

Рис. 135. Повідомлення про успішне створення таблиці **employee**

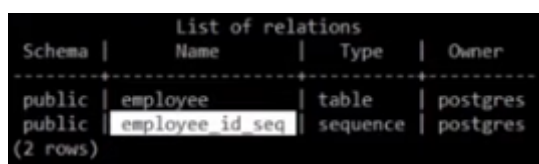
Після внесення команди **avecoder=# \d**, щоб подивитися список таблиць і спостерігаємо запис про новостворену таблицю (рис. 136).



```
avecoder=# \d
List of relations
Schema | Name | Type | Owner
-----+-----+-----+-----
public | employee | table | postgres
public | employee_id_seq | sequence | postgres
(2 rows)
```

Рис. 136. Перевірка новоствореної таблиці **employee** у переліку таблиць

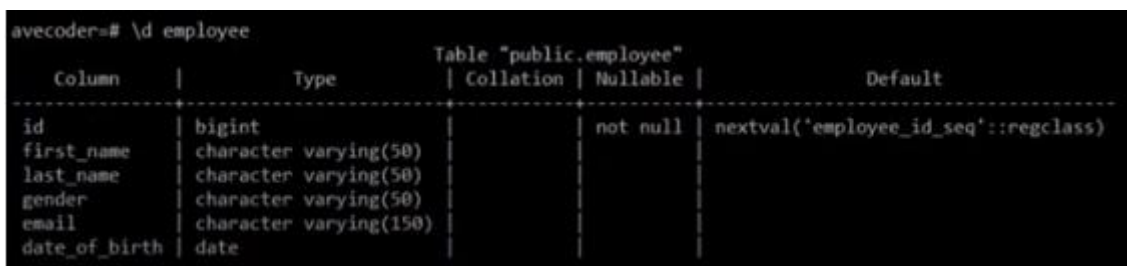
І другий запис про **id**, тобто в третій колонці навпроти **employee_id_seq** вказано тип **sequence** – послідовність, а не **table** (рис. 137).



```
List of relations
Schema | Name | Type | Owner
-----+-----+-----+-----
public | employee | table | postgres
public | employee_id_seq | sequence | postgres
(2 rows)
```

Рис. 137. Тип **sequence** – послідовність

Подивимося, що є в таблиці **employee** (рис. 138).



```
avecoder=# \d employee
Table "public.employee"
Column | Type | Collation | Nullable | Default
-----+-----+-----+-----+-----
id | bigint | | not null | nextval('employee_id_seq'::regclass)
first_name | character varying(50) | | | 
last_name | character varying(50) | | | 
gender | character varying(50) | | | 
email | character varying(150) | | | 
date_of_birth | date | | |
```

Рис. 138. Зміст таблиці **employee**

2. Видалення таблиці – **DROP TABLE** (рис. 139).

```
avecoder=# DROP TABLE employee;
DROP TABLE
avecoder=# \d employee
Did not find any relation named "employee".
```

Рис. 139. Видалення таблиці **employee**

Заново перестворимо таблицю, тільки додамо обмеження:

- **id**, **first_name**, **last_name**, **gender** та **date_of_birth** мають обмеження **NOT NULL**, а **id** – первинний ключ (**PRIMARY KEY**) (рис. 140).

```
SQL Shell (psql)
avecoder=# CREATE TABLE employee (
avecoder(# id BIGSERIAL NOT NULL PRIMARY KEY,
avecoder(# first_name VARCHAR(50) NOT NULL,
avecoder(# last_name VARCHAR(50) NOT NULL,
avecoder(# gender VARCHAR(6) NOT NULL,
avecoder(# email VARCHAR(150),
avecoder(# date_of_birth DATE NOT NULL
avecoder(# );
```

Рис. 140. Перестворення таблиці **employee** з додаванням обмеження **NOT NULL** та первинного ключа **PRIMARY KEY**

Після запуску отримуємо **CREATE TABLE** та перевіряємо **\d** (рис. 141).

```
CREATE TABLE
avecoder=# \d
List of relations
Schema | Name | Type | Owner
-----+-----+-----+-----
public | employee | table | postgres
public | employee_id_seq | sequence | postgres
(2 rows)
```

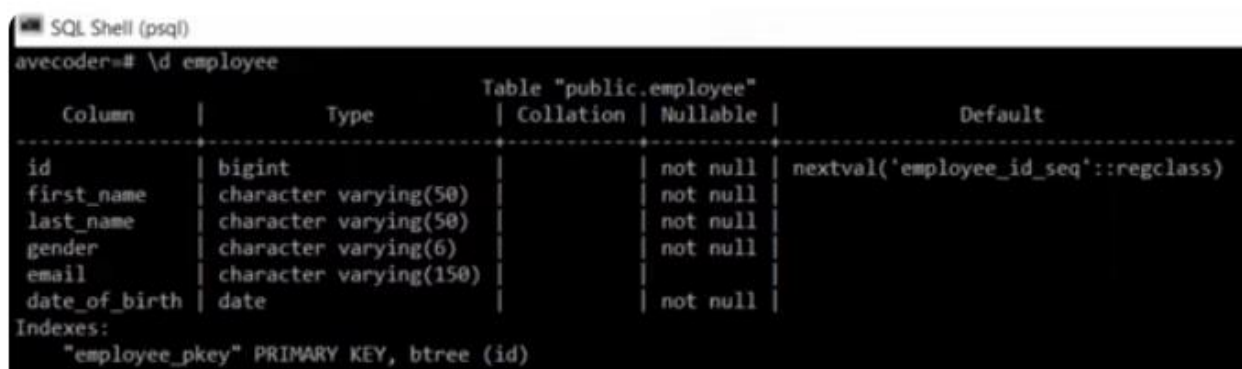
Рис. 141. Перевірка переліку створених таблиць в базі даних

Подивимося поля таблиці **\d employee** (рис. 142).

```
avecoder=# \d employee
Table "public.employee"
Column | Type | Collation | Nullable | Default
-----+-----+-----+-----+-----
id | bigint | | not null | nextval('employee_id_seq'::regclass)
first_name | character varying(50) | | not null |
last_name | character varying(50) | | not null |
gender | character varying(6) | | not null |
email | character varying(150) | | not null |
date_of_birth | date | | not null |
Indexes:
"employee_pkey" PRIMARY KEY, btree (id)
```

Рис. 142. Структура створеної таблиці **employee**

3. Заповнення таблиці **employee** даними (рис. 143).



```
SQL Shell (psql)
avecoder=# \d employee
```

Column	Type	Table "public.employee"	Collation	Nullable	Default
id	bigint			not null	nextval('employee_id_seq'::regclass)
first_name	character varying(50)			not null	
last_name	character varying(50)			not null	
gender	character varying(6)			not null	
email	character varying(150)			not null	
date_of_birth	date			not null	

Indexes:
"employee_pkey" PRIMARY KEY, btree (id)

Рис. 143. Заповнення таблиці **employee** даними

Заповнення будемо проводити за допомогою команди **INSERT INTO** із зазначенням полів та їхніх значень (рис. 144).

```
avecoder=# INSERT INTO employee (
avecoder(# first_name, last_name, gender, email, date_of_birth)
avecoder-# VALUES('John', 'Doe', 'MALE', 'Jd@mail.com', '2000-01-01');
```

Рис. 144. Заповнення таблиці **employee** даними за допомогою команди **INSERT INTO**

Після виконання, отримуємо повідомлення про вдале виконання **INSERT 0 1**. Перевіримо, що кількість записів не 0 (рис. 145).

```
avecoder=# \dt employee
```

Schema	Name	Type	Owner
public	employee	table	postgres

(1 row)

Рис. 145. Перевірка кількості записів

Контрольні запитання та завдання

1. Чим відрізняється однорядковий оператор **INSERT** від багаторядкового оператора **INSERT**?
2. За допомогою яких інструкцій здійснюється створення таблиці бази даних засобами СУБД PostgreSQL (pgAdmin)?
3. За допомогою яких інструкцій здійснюється створення таблиці бази даних засобами СУБД PostgreSQL (SQL Shell)?
4. За допомогою яких інструкцій здійснюється створення таблиці бази даних засобами СУБД MySQL?

ЛАБОРАТОРНА РОБОТА № 5

ВІДПРАЦЮВАННЯ ПОШУКУ ДАНИХ У СЕРЕДОВИЩІ СИСТЕМИ УПРАВЛІННЯ БАЗАМИ ДАНИХ POSTGRESQL

Мета:

Навчитися створювати та реалізовувати SQL-запити у середовищі системи управління базами даних PostgreSQL.

Завдання:

1. Які компанії виробники є в базі?
2. У яких містах є склади?
3. Знайти солодоші з вичерпаним терміном придатності.
4. Знайти солодоші, у яких вартість від 20 до 300 умовних одиниць.
5. Знайти солодоші, у яких назва починається на букву Ш.
6. Скласти список солодошів, відсортованих від А до Я
7. Знайти види солодошів, у яких кількість більша за 2 одиниць.
8. Яке максимальне значення ідентифікатора у солодошів?

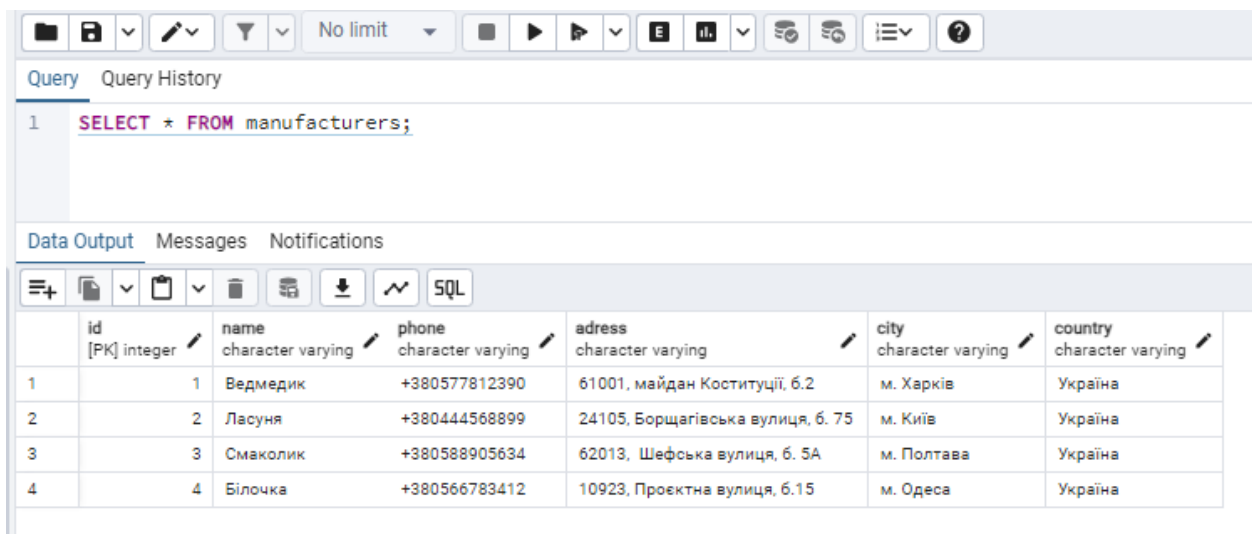
Виконання завдання 1

Які компанії виробники є в базі?

Рішення:

```
SELECT * FROM manufacturers;
```

Вивантажте всі стовпці з таблиці **manufacturers** (рис. 146).



	id [PK] integer	name character varying	phone character varying	adress character varying	city character varying	country character varying
1	1	Ведмедик	+380577812390	61001, майдан Коституції, б.2	м. Харків	Україна
2	2	Ласуня	+380444568899	24105, Борщагівська вулиця, б. 75	м. Київ	Україна
3	3	Смаколик	+380588905634	62013, Шефська вулиця, б. 5А	м. Полтава	Україна
4	4	Білочка	+380566783412	10923, Проектна вулиця, б.15	м. Одеса	Україна

Рис. 146. Результат виконання запиту на вивантаження всіх стовпців з таблиці **manufacturers**

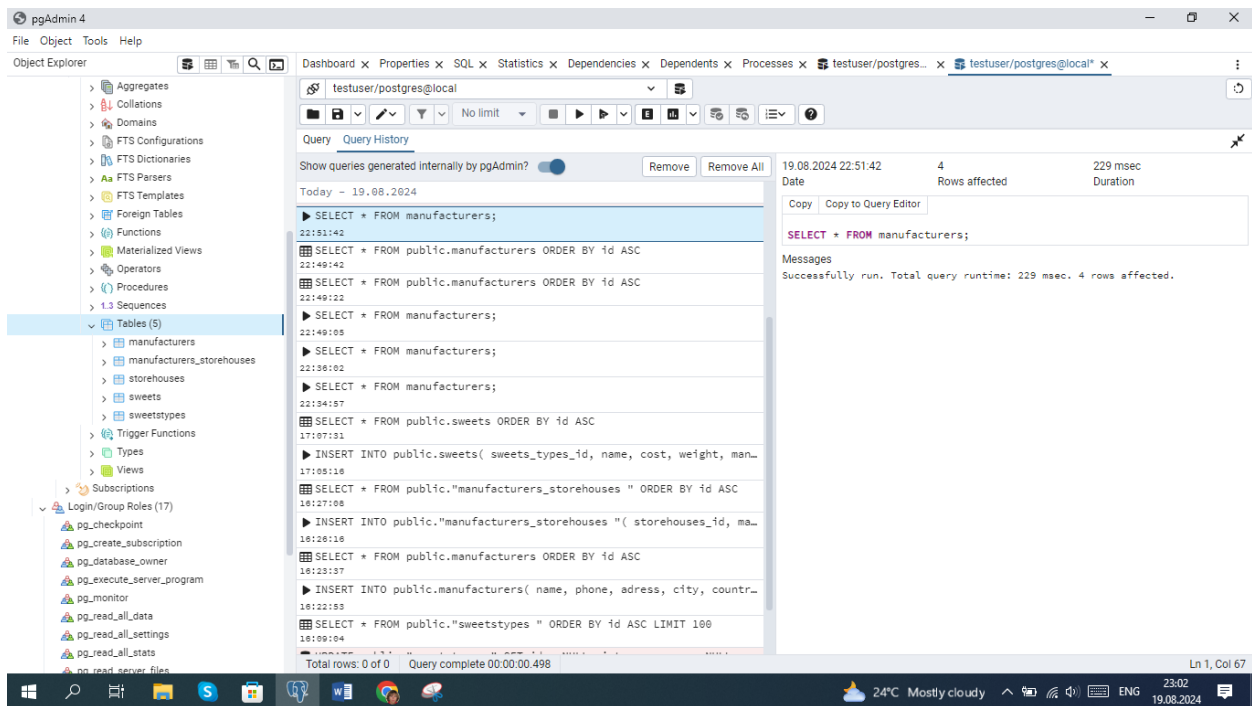


Рис. 146. Результат виконання запиту на вивантаження всіх стовпців з таблиці **manufacturers** (закінчення)

Виконання завдання 2 У яких містах є склади?

Рішення:

SELECT DISTINCT city FROM storehouses (рис. 147).

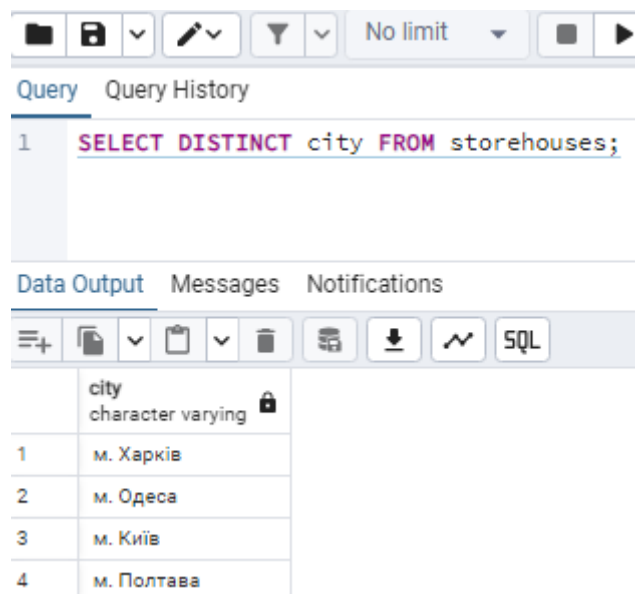


Рис. 147. Результат виконання запиту на обрання записів, які містять інформацію про міста, де є склади

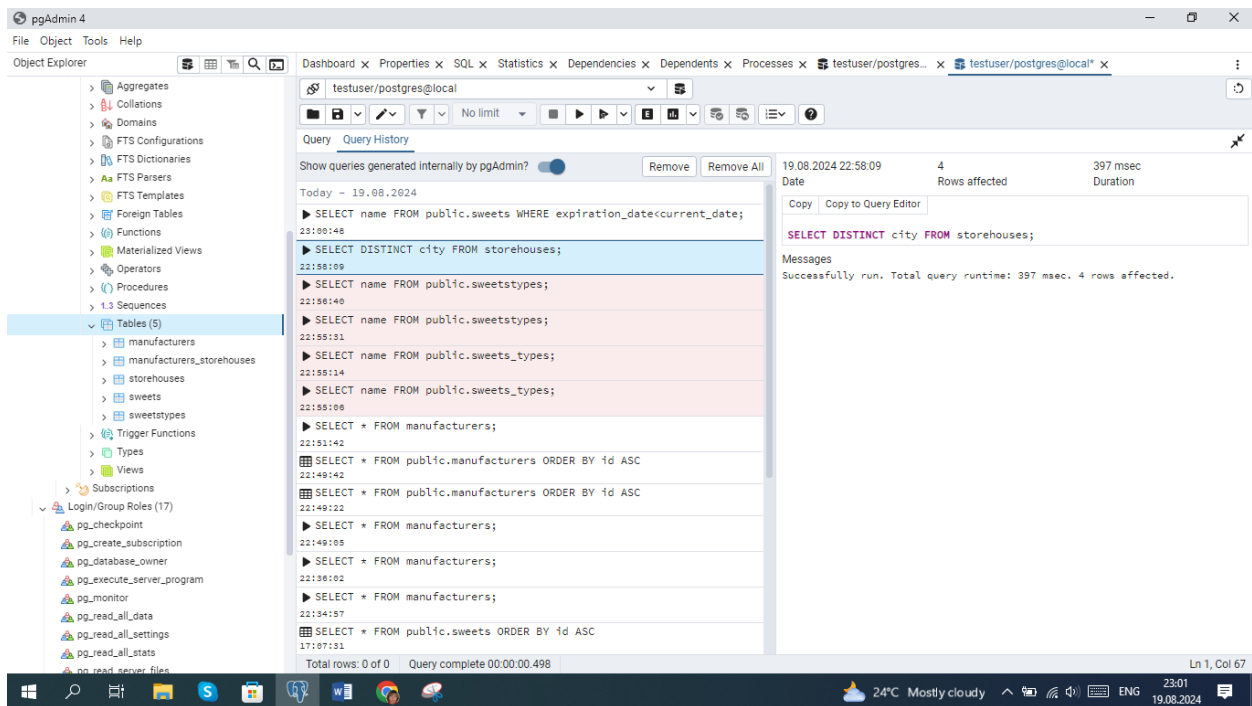


Рис. 147. Результат виконання запиту на обрання записів, які містять інформацію про міста, де є склади (закінчення)

Виконання завдання 3

Знайти солодощі з вичерпаним терміном придатності

Підказка: використовуйте для умови змінну `current_date`.

Рішення:

SELECT name **FROM** public.sweets **WHERE**
expiration_date<current_date (рис. 148).

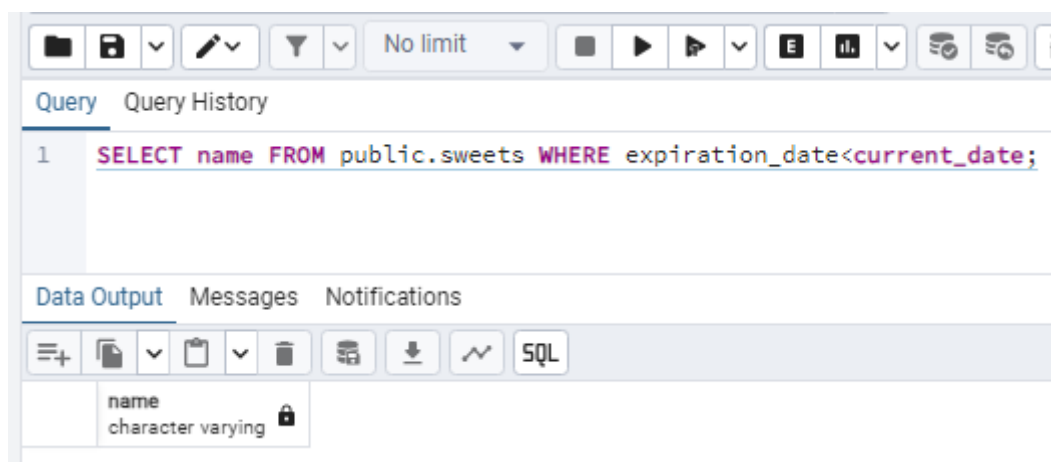


Рис. 148. Результат виконання запиту на знаходження записів із солодощами з вичерпаним терміном придатності

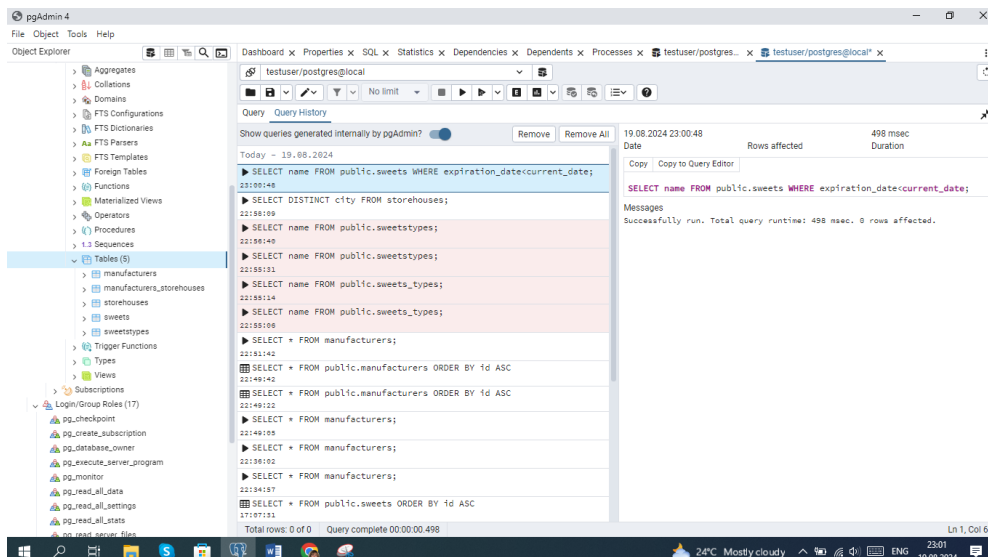


Рис. 148. Результат виконання запиту на знаходження записів із солодощами з вичерпаним терміном придатності (закінчення)

Виконання завдання 4

Знайти солодощі, у яких вартість від 20 до 300 умовних одиниць

Рішення:

SELECT * FROM public.sweets WHERE cost >= '20' AND cost < '300'
(рис. 149).

Query										
1 SELECT * FROM public.sweets WHERE cost >= '20' AND cost < '300';										
Data Output										
	id [PK] integer	sweets_types_id integer	name character varying	cost character varying	weight character varying	manufacturers_id integer	with_sugar boolean	requires_freezing boolean	production_date date	expiration_date date
1	1	1	Апрек	30	1kr		1	false	2024-09-03	2025-03-15

Рис. 149. Результат виконання запиту на знаходження записів із солодощами, у яких вартість від 20 до 300 умовних одиниць

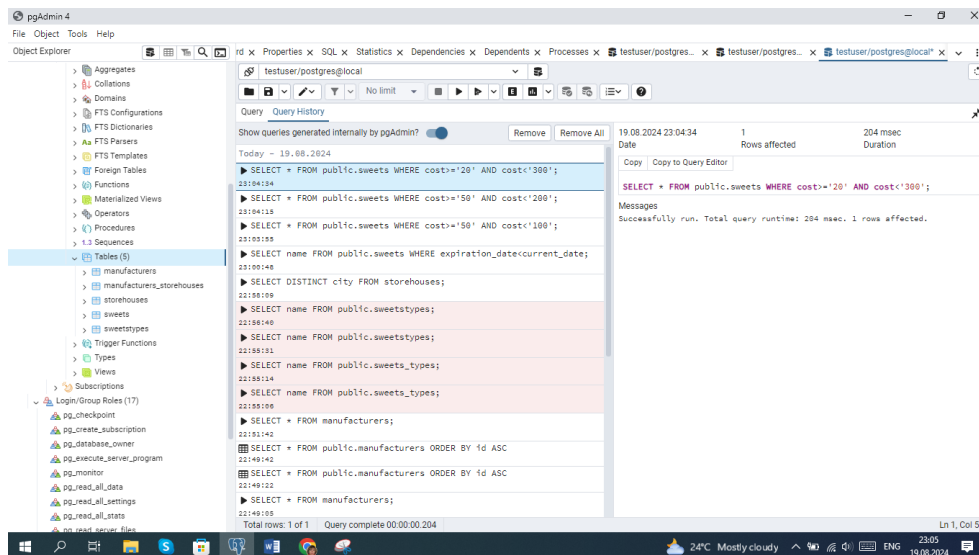


Рис. 149. Результат виконання запиту на знаходження записів із солодощами, у яких вартість від 20 до 300 умовних одиниць (закінчення)

або

SELECT * FROM public.sweets WHERE cost BETWEEN '20' AND '300';(рис. 150);

Query										
1 SELECT * FROM public.sweets WHERE cost BETWEEN '20' AND '300';										
Data Output										
	id [PK] integer	sweets_types_id integer	name character varying	cost character varying	weight character varying	manufacturers_id integer	with_sugar boolean	requires_freezing boolean	production_date date	expiration_date date
1	1	1	Аптек	30	1kr		1	false	2024-09-03	2025-03-15

Рис. 150. Результат виконання запиту на знаходження записів із солодощами, у яких вартість від 20 до 300 умовних одиниць (другий спосіб)

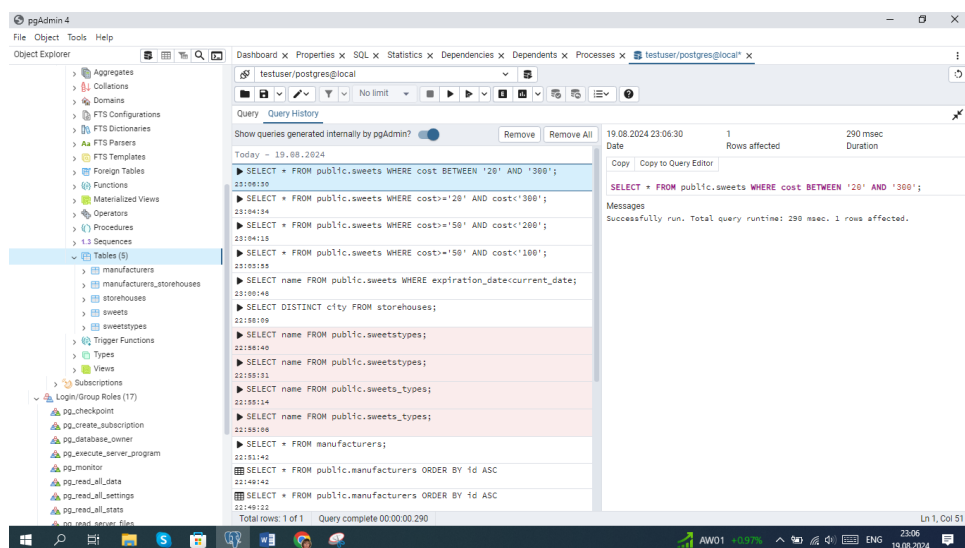


Рис. 150. Результат виконання запиту на знаходження записів із солодощами, у яких вартість від 20 до 300 умовних одиниць (другий спосіб) (закінчення)

Оскільки в умові завдання не сказано, чи потрібно включати до вибірки вартість, що дорівнює 20 і дорівнює 300, то запит

SELECT * FROM public.sweets WHERE cost BETWEEN '20' AND '300'
дасть включно з вартістю 20 і 300,

а запит

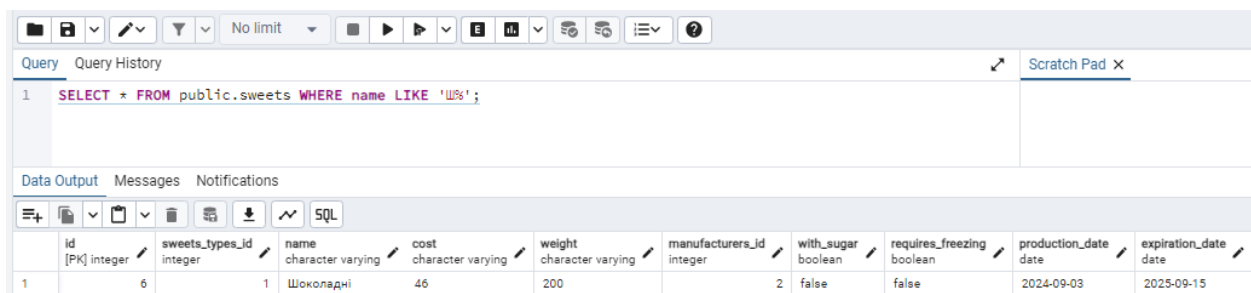
SELECT * FROM public.sweets WHERE cost >='20' AND cost <'300',
не включить вартість 300.

Виконання завдання 5

Знайти солодощі, у яких назва починається на букву Ш

Рішення:

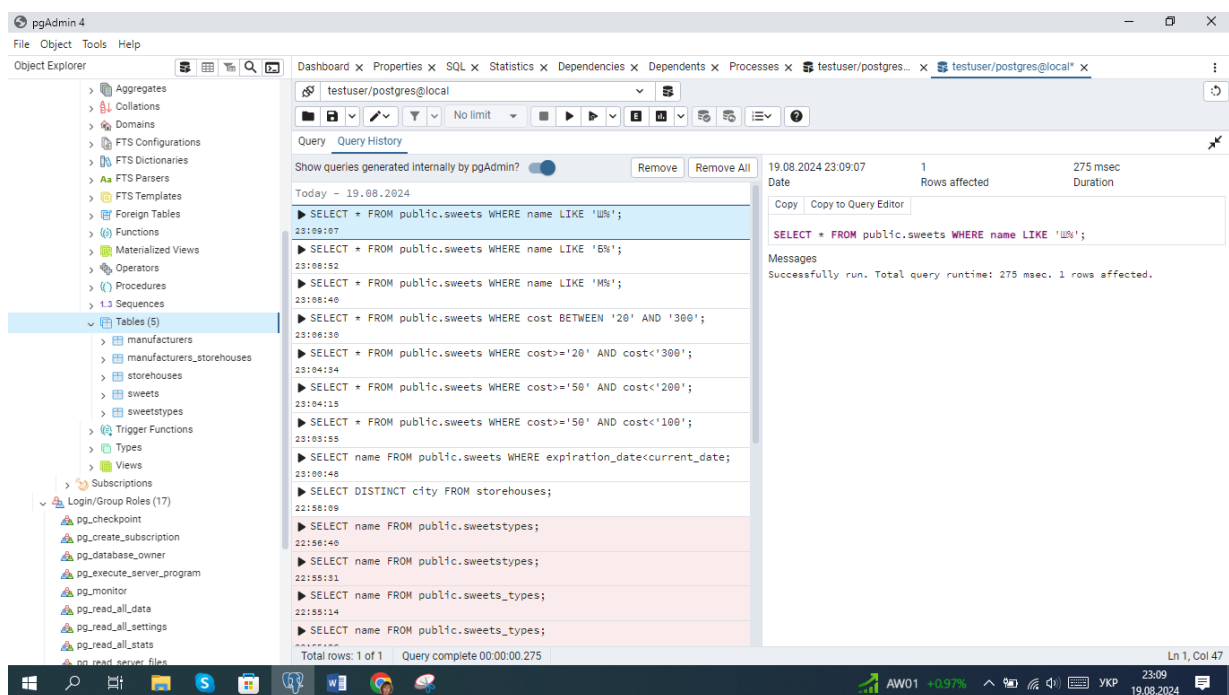
SELECT * FROM public.sweets WHERE name LIKE 'Ш%' (рис. 151).



The screenshot shows a SQL query editor with the following query: `SELECT * FROM public.sweets WHERE name LIKE 'Ш%';`. Below the query, the 'Data Output' tab is active, displaying a table with 10 columns: id, sweets_types_id, name, cost, weight, manufacturers_id, with_sugar, requires_freezing, production_date, and expiration_date. The first row of data is shown.

id	sweets_types_id	name	cost	weight	manufacturers_id	with_sugar	requires_freezing	production_date	expiration_date
1	6	Шоколадні	46	200	2	false	false	2024-09-03	2025-09-15

Рис. 151. Результат виконання запиту на знаходження записів із солодощами, у яких назва починається на букву Ш



The screenshot shows the pgAdmin 4 interface. The 'Query History' tab is active, displaying a list of queries executed. The first query is highlighted, showing its execution details: Date: 19.08.2024 23:09:07, Rows affected: 1, Duration: 275 msec. The query text is: `SELECT * FROM public.sweets WHERE name LIKE 'Ш%';`. The 'Messages' tab shows the success message: 'Successfully run. Total query runtime: 275 msec. 1 rows affected.'

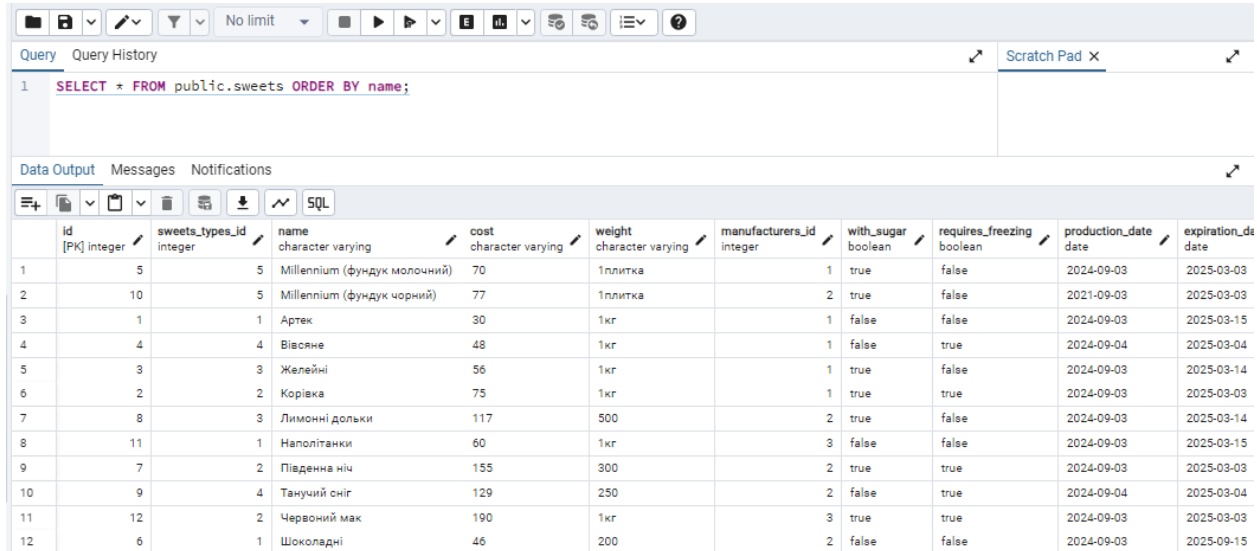
Рис. 151. Результат виконання запиту на знаходження записів із солодощами, у яких назва починається на букву Ш (закінчення)

Виконання завдання 6

Скласти список солодошів, відсортованих від А до Я

Рішення:

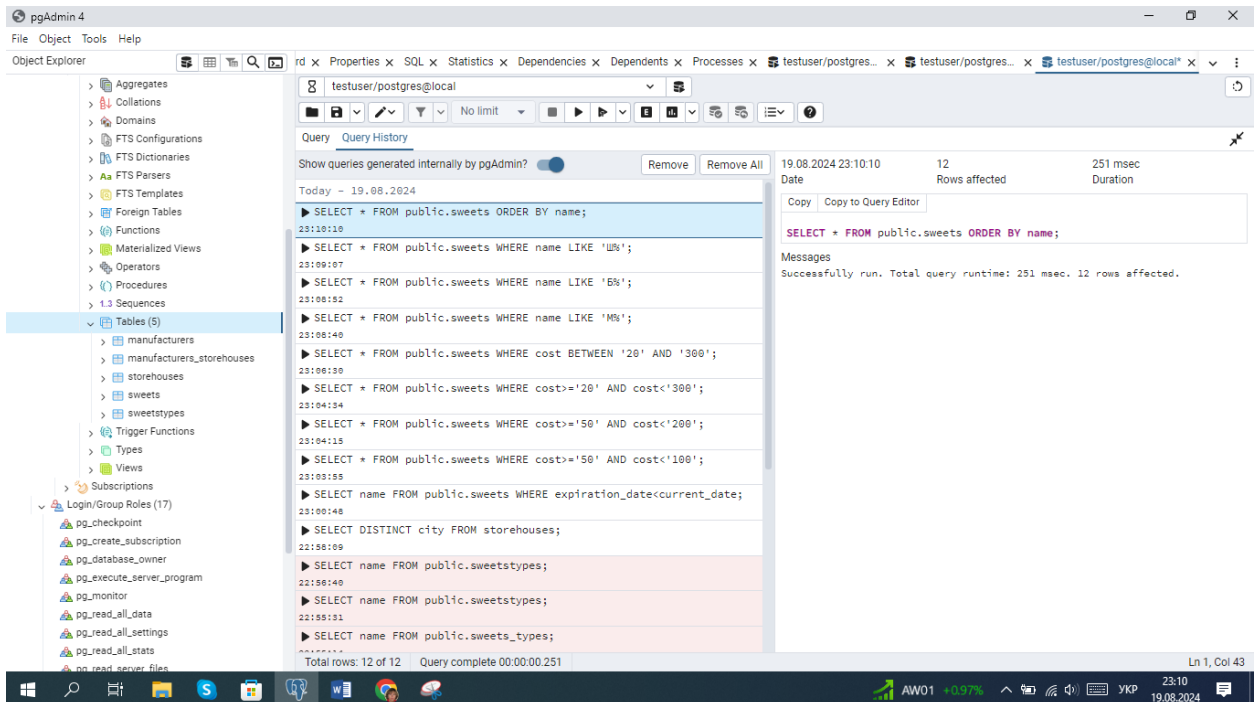
SELECT * FROM public.sweets ORDER BY name (рис. 152).



The screenshot shows a database query tool interface. The query editor at the top contains the SQL statement: `SELECT * FROM public.sweets ORDER BY name;`. Below the query editor, the 'Data Output' tab is active, displaying a table with 12 rows and 12 columns. The columns are: id [PK] integer, sweets_types_id integer, name character varying, cost character varying, weight character varying, manufacturers_id integer, with_sugar boolean, requires_freezing boolean, production_date date, and expiration_date date. The rows are sorted alphabetically by the 'name' column.

	id [PK] integer	sweets_types_id integer	name character varying	cost character varying	weight character varying	manufacturers_id integer	with_sugar boolean	requires_freezing boolean	production_date date	expiration_date date
1	5	5	Millennium (фундук молочний)	70	1плитка	1	true	false	2024-09-03	2025-03-03
2	10	5	Millennium (фундук чорний)	77	1плитка	2	true	false	2021-09-03	2025-03-03
3	1	1	Артек	30	1кг	1	false	false	2024-09-03	2025-03-15
4	4	4	Віссяне	48	1кг	1	false	true	2024-09-04	2025-03-04
5	3	3	Желейні	56	1кг	1	true	false	2024-09-03	2025-03-14
6	2	2	Корівка	75	1кг	1	true	true	2024-09-03	2025-03-03
7	8	3	Лимонні дольки	117	500	2	true	false	2024-09-03	2025-03-14
8	11	1	Наполітани	60	1кг	3	false	false	2024-09-03	2025-03-15
9	7	2	Південна ніч	155	300	2	true	true	2024-09-03	2025-03-03
10	9	4	Танцюючий сніг	129	250	2	false	true	2024-09-04	2025-03-04
11	12	2	Червоний мак	190	1кг	3	true	true	2024-09-03	2025-03-03
12	6	1	Шоколадні	46	200	2	false	false	2024-09-03	2025-09-15

Рис. 152. Результат виконання запиту на знаходження записів зі складання списку солодошів, відсортованих від А до Я



The screenshot shows the pgAdmin 4 interface. The 'Query History' tab is active, displaying a list of queries. The first query is the same SELECT statement: `SELECT * FROM public.sweets ORDER BY name;`. The status bar at the bottom indicates that the query was successfully run, affecting 12 rows in 251 msec.

Query	Date	Rows affected	Duration
SELECT * FROM public.sweets ORDER BY name;	19.08.2024 23:10:10	12	251 msec

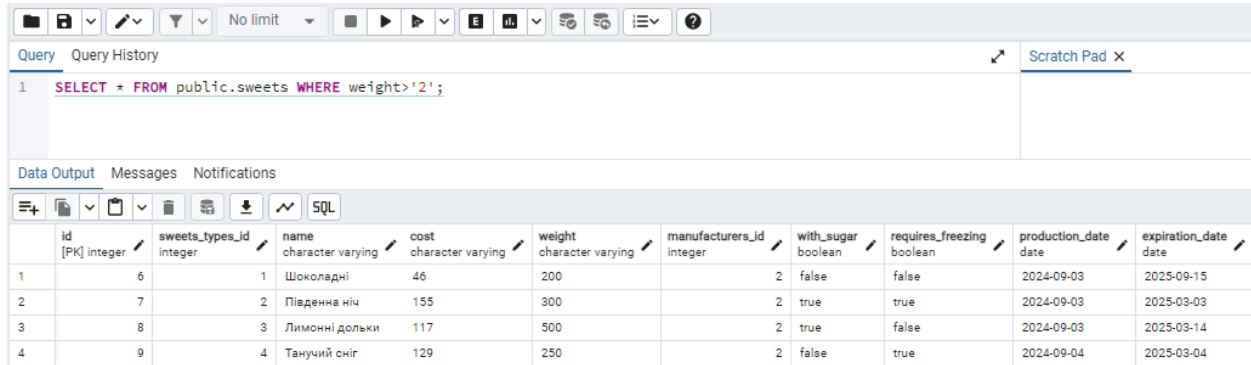
Рис. 152. Результат виконання запиту на знаходження записів зі складання списку солодошів, відсортованих від А до Я (закінчення)

Виконання завдання 7

Знайти види солодощів, у яких кількість більша за 2 одиниці

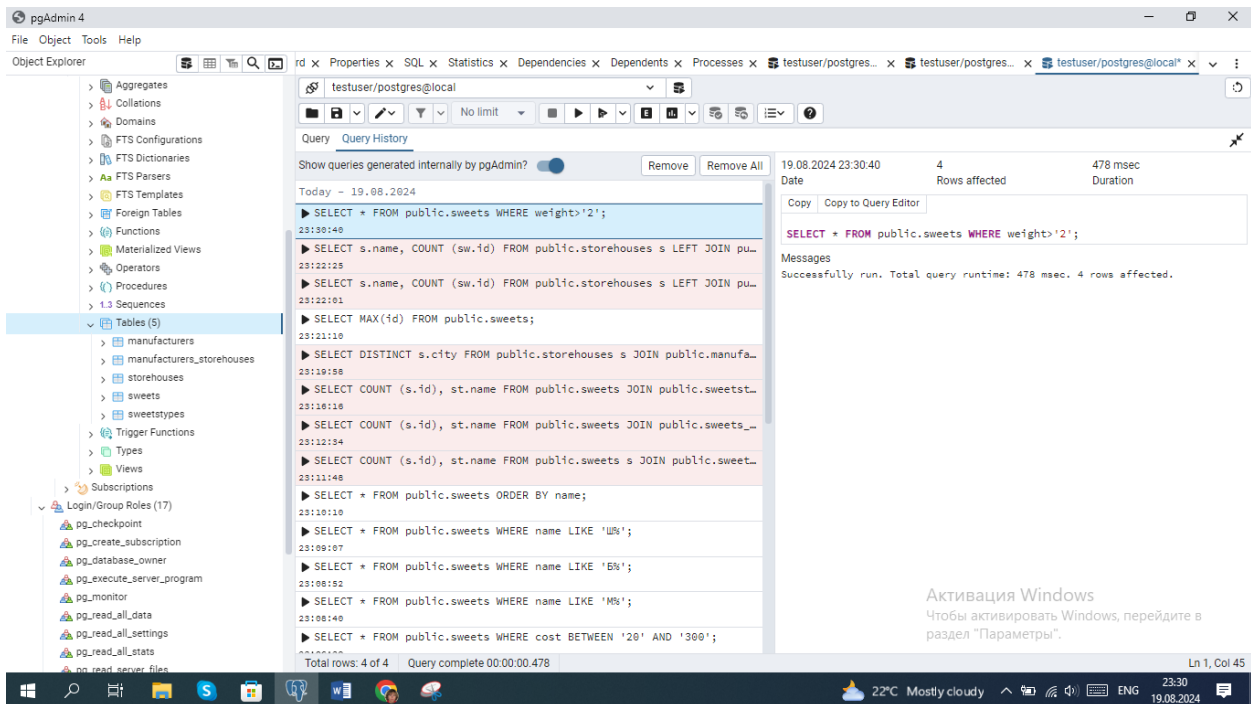
Рішення:

SELECT * FROM public.sweets WHERE weight>'2' (рис. 153).



id	sweets_types_id	name	cost	weight	manufacturers_id	with_sugar	requires_freezing	production_date	expiration_date
1	6	Шоколадні	46	200	2	false	false	2024-09-03	2025-09-15
2	7	Півадна ніч	155	300	2	true	true	2024-09-03	2025-09-03
3	8	Лимонні дольки	117	500	2	true	false	2024-09-03	2025-03-14
4	9	Танучий сніг	129	250	2	false	true	2024-09-04	2025-03-04

Рис. 153. Результат виконання запиту на знаходження записів із видів солодощів, кількість яких більша за 2 одиниці



pgAdmin 4

Object Explorer

- Aggregates
- Collations
- Domains
- FTS Configurations
- FTS Dictionaries
- FTS Parsers
- FTS Templates
- Foreign Tables
- Functions
- Materialized Views
- Operators
- Procedures
- Sequences
- Tables (5)
 - manufacturers
 - manufacturers_storehouses
 - storehouses
 - sweets
 - sweetstypes
- Trigger Functions
- Types
- Views
- Subscriptions
- Login/Group Roles (17)
 - pg_checkpoint
 - pg_create_subscription
 - pg_database_owner
 - pg_execute_server_program
 - pg_monitor
 - pg_read_all_data
 - pg_read_all_settings
 - pg_read_all_stats
 - pg_read_server_files

Query History

Show queries generated internally by pgAdmin? ☒ Remove Remove All

Today - 19.08.2024

- 23:30:40
▶ SELECT * FROM public.sweets WHERE weight>'2';
- 23:22:01
▶ SELECT s.name, COUNT (sw.id) FROM public.storehouses s LEFT JOIN pu...
- 23:22:01
▶ SELECT s.name, COUNT (sw.id) FROM public.storehouses s LEFT JOIN pu...
- 23:22:01
▶ SELECT MAX(id) FROM public.sweets;
- 23:21:10
▶ SELECT DISTINCT s.city FROM public.storehouses s JOIN public.manufa...
- 23:19:58
▶ SELECT COUNT (s.id), st.name FROM public.sweets JOIN public.sweetst...
- 23:16:16
▶ SELECT COUNT (s.id), st.name FROM public.sweets JOIN public.sweetst...
- 23:12:54
▶ SELECT COUNT (s.id), st.name FROM public.sweets s JOIN public.sweet...
- 23:11:48
▶ SELECT * FROM public.sweets ORDER BY name;
- 23:10:10
▶ SELECT * FROM public.sweets WHERE name LIKE 'Ш%';
- 23:09:07
▶ SELECT * FROM public.sweets WHERE name LIKE 'Б%';
- 23:08:52
▶ SELECT * FROM public.sweets WHERE name LIKE 'М%';
- 23:08:40
▶ SELECT * FROM public.sweets WHERE cost BETWEEN '20' AND '300';

Total rows: 4 of 4 Query complete 00:00:00.478

19.08.2024 23:30:40 4 Rows affected 478 msec

Copy Copy to Query Editor

SELECT * FROM public.sweets WHERE weight>'2';

Messages

Successfully run. Total query runtime: 478 msec. 4 rows affected.

Активация Windows
Чтобы активировать Windows, перейдите в раздел "Параметры".

Ln 1, Col 45

Рис. 153. Результат виконання запиту на знаходження записів із видів солодощів, кількість яких більша за 2 одиниці (закінчення)

Виконання завдання 8

Яке максимальне значення ідентифікатора у солодошів?

Рішення:

SELECT MAX(id) FROM public.sweets (рис. 154).

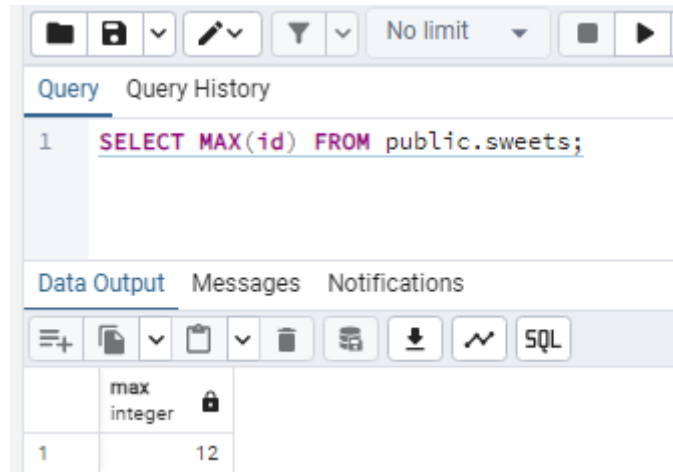


Рис. 154. Результат виконання запиту на знаходження записів із максимальним значенням ідентифікатора у солодошів

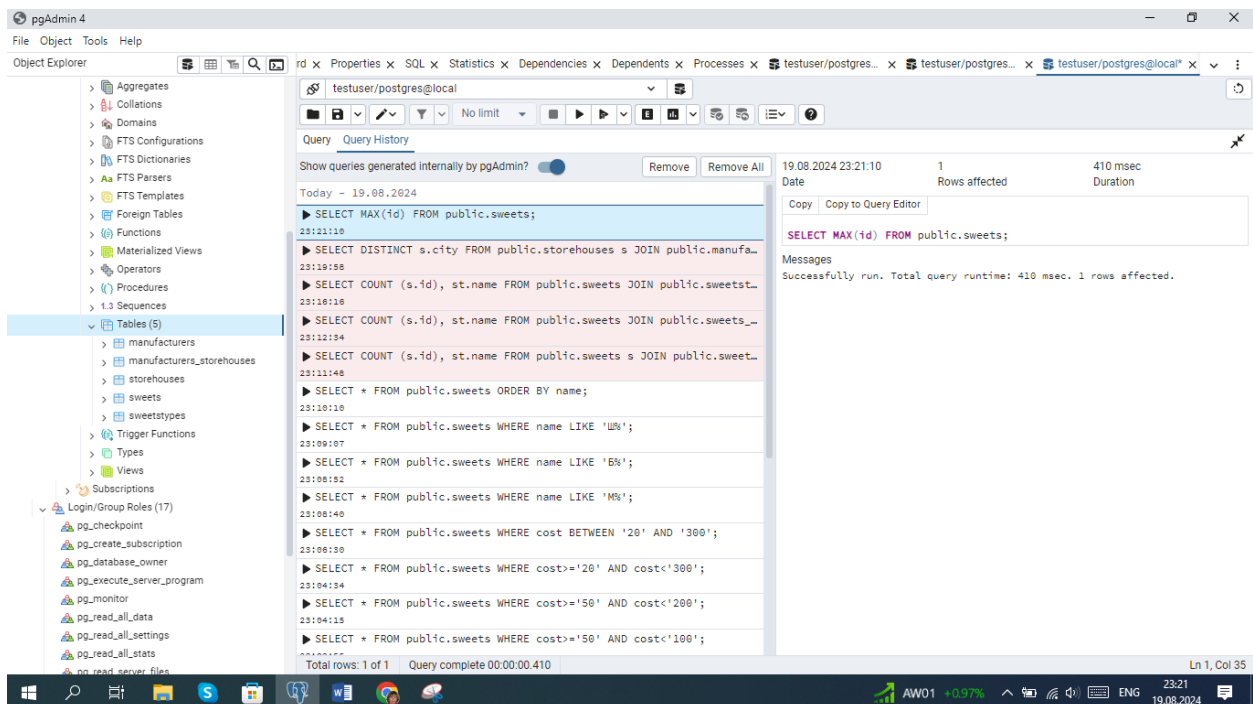


Рис. 154. Результат виконання запиту на знаходження записів із максимальним значенням ідентифікатора у солодошів (закінчення)

Контрольні запитання та завдання

1. Яка інструкція реалізує обрання всіх записів таблиці бази даних?
2. Яка інструкція реалізує обрання міст, в яких є склади.?
3. Яка інструкція реалізує обрання солодоців з вичерпаним терміном придатності?
4. Яка інструкція реалізує обрання солодоців, вартість яких варіюється у певному інтервалі умовних одиниць?
5. Яка інструкція реалізує обрання солодоців, назва яких починається на певну букву алфавіту?
6. Яка інструкція реалізує складання списку солодоців, відсортованих за алфавітом у порядку зростання?
7. Яка інструкція реалізує пошук видів солодоців, у яких кількість більша за встановлене значення?
8. Яка інструкція реалізує обрання максимального значення ідентифікатора у солодоців.

ЛАБОРАТОРНА РОБОТА № 6

РЕАЛІЗАЦІЯ КОМАНД ЗАПИТІВ SQL, ОБМЕЖЕНЬ ТА ПЕРЕВІРКИ ЗА ДОПОМОГОЮ SQL SHELL (КОНСОЛІ)

Мета:

Придбання навичок щодо створення та реалізації запитів SQL засобами СУБД PostgreSQL

Завдання:

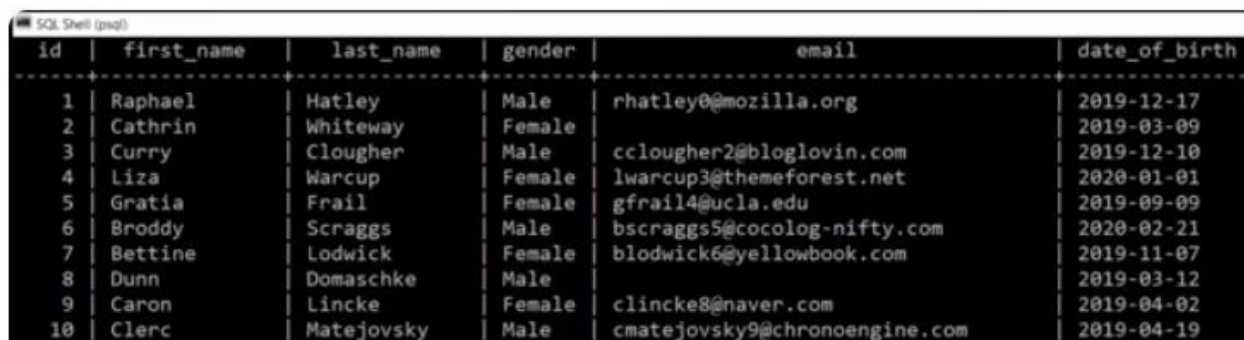
1. Реалізація команд запитів SQL: SELECT, ORDER BY, DISTINCT, WHERE, AND, OR, LIMIT, OFFSET, FETCH.
2. Реалізація команд запитів SQL: IN, BETWEEN, LIKE, iLIKE, GROUP BY, HAVING, ALIAS, COALESCE.
3. Реалізація обмеження та перевірки – UNIQUE, CHECK.

Виконання завдання 1

Реалізація команд запитів SQL: SELECT, ORDER BY, DISTINCT, WHERE, AND, OR, LIMIT, OFFSET, FETCH

Сформувавши SQL-запит на виведення всього вмісту таблиці

`sqlcoder=# SELECT * FROM employee;`, отримаємо введені записи (рис. 155).



id	first_name	last_name	gender	email	date_of_birth
1	Raphael	Hatley	Male	rhatley0@mozilla.org	2019-12-17
2	Cathrin	Whiteway	Female		2019-03-09
3	Curry	Clougher	Male	ccclougher2@bloglovin.com	2019-12-10
4	Liza	Warcup	Female	lwarcup3@theforest.net	2020-01-01
5	Gratia	Frail	Female	gfrail4@ucla.edu	2019-09-09
6	Broddy	Scraggs	Male	bbscrags5@cocolog-nifty.com	2020-02-21
7	Bettine	Lodwick	Female	blodwick6@yellowbook.com	2019-11-07
8	Dunn	Domaschke	Male		2019-03-12
9	Caron	Lincke	Female	clincke8@naver.com	2019-04-02
10	Clerc	Matejovsky	Male	cmatejovsky9@chronoengine.com	2019-04-19

Рис. 155. Результат виконання команди запиту **SELECT**

При внесенні наступного запиту, отримаємо загальну кількість записів у таблиці **employee** (рис. 156).

```
SQL Shell (psql)
avecoder=# SELECT FROM employee;
--
(1000 rows)
```

Рис. 156. Результат виконання запиту з отримання загальної кількості записів таблиці **employee**

Отримання переліку внесених імен співробітників (рис. 157).

```
avecoder=# SELECT first_name FROM employee;
```

first_name
Raphael
Cathrin
Curry
Liza
Gratia
Broddy
Bettine
Dunn
Caron
Clerc

Рис. 157. Результат виконання запиту з отримання переліку внесених імен співробітників в таблиці **employee**

Додаємо ще одну колонку з прізвищами (last_name) (рис. 158).

```
SQL Shell (psql)
avecoder=# SELECT first_name, last_name FROM employee;
```

first_name	last_name
Raphael	Hatley
Cathrin	Whiteway
Curry	Clougher
Liza	Warcup
Gratia	Frail
Broddy	Scraggs
Bettine	Lodwick
Dunn	Domaschke
Caron	Lincke
Clerc	Matejovsky

Рис. 158. Результат додавання колонки last_name в таблицю **employee**

Виконання запиту на вибір даних із поля email (рис. 159).

```
SQL Shell (psql)
avecoder=# SELECT email FROM employee;

SQL Shell (psql)

      email
-----
rhatley0@mozilla.org

cclougher2@bloglovin.com
lwarcup3@theforest.net
gfrail4@ucla.edu
bscraggs5@cocolog-nifty.com
blodwick6@yellowbook.com

clincke8@naver.com
cmatejovsky9@chronoengine.com
gbettesona@virginia.edu
kivanonkob@geocities.com
```

Рис. 159. Результат виконання запиту на вибір даних із поля email таблиці **employee**

Частину адрес не буде заповнено, оскільки у цього поля немає обмеження на відсутність нульових значень (рис. 160).

```
Select SQL Shell (psql)

      email
-----
rhatley0@mozilla.org
cclougher2@bloglovin.com
lwarcup3@theforest.net
gfrail4@ucla.edu
bscraggs5@cocolog-nifty.com
blodwick6@yellowbook.com
```

Рис. 160. Результат виконання запиту на вибір даних із поля email таблиці **employee** без обмеження нульових значень

Вибір з умовою вибірки за зростанням прізвищ (last_name) за допомогою команди **ORDER BY** (рис. 161).

```
avecoder=# SELECT * FROM employee ORDER BY last_name;
```

id	first_name	last_name	gender	email	date_of_birth
1	Jone	Doe	MALE	Jd@gmail.com	2024-09-21
2	Jone	Key	MALE	kre@gmail.com	2014-09-21
3	Konst	Lyman	MALE	dpr@gmail.com	2015-09-21

Рис. 161. Результат виконання запиту з умовою вибірки за зростанням прізвищ (last_name) за допомогою команди **ORDER BY** з таблиці **employee**

Вибір з умовою вибірки за спаданням прізвищ (last_name) за допомогою команди **DESC** (рис. 162).

```
avecoder=# SELECT * FROM employee ORDER BY last_name DESC;
```

id	first_name	last_name	gender	email	date_of_birth
3	Konst	Lyman	MALE	dpr@gmail.com	2015-09-21
2	Jone	Key	MALE	kre@gmail.com	2014-09-21
1	Jone	Doe	MALE	Jd@gmail.com	2024-09-21

Рис. 162. Результат виконання запиту з умовою вибірки за спаданням прізвищ (last_name) за допомогою команди **DESC** з таблиці **employee**

Вибір з умовою вибірки за зростанням електронних адрес (email) з результатом (записи з нульовими значеннями поля розміщуватимуться в кінці списку, якщо такі є) (рис. 163).

```
avecoder=# SELECT * FROM employee ORDER BY email;
```

id	first_name	last_name	gender	email	date_of_birth
3	Konst	Lyman	MALE	dpr@gmail.com	2015-09-21
1	Jone	Doe	MALE	Jd@gmail.com	2024-09-21
2	Jone	Key	MALE	kre@gmail.com	2014-09-21

Рис. 163. Результат виконання запиту з умовою вибірки за зростанням електронних адрес (email) з результатом (записи з нульовими значеннями поля розміщуватимуться в кінці списку, якщо такі є) з таблиці **employee**

Вибір з умовою вибірки за зменшенням електронних адрес (email) і результат виконання запиту (спочатку будуть слідувати email-ли з нульовими значеннями, якщо такі є) (рис. 164).

```
avecoder=# SELECT * FROM employee ORDER BY email DESC;
```

id	first_name	last_name	gender	email	date_of_birth
2	Jone	Key	MALE	kre@gmail.com	2014-09-21
1	Jone	Doe	MALE	Jd@gmail.com	2024-09-21
3	Konst	Lyman	MALE	dpr@gmail.com	2015-09-21

Рис. 164. Результат виконання запиту з умовою вибірки за зменшенням електронних адрес (email) і результат виконання запиту (спочатку будуть слідувати email-ли з нульовими значеннями, якщо такі є) з таблиці **employee**

Вибір прізвищ під час сортування за прізвищем – при великій кількості записів у цьому разі спочатку впорядковуються записи за одним прізвищем, потім за іншим і т.д. (з одним і тим же прізвищем) (рис. 165).

```

avocoder=# SELECT last_name FROM employee ORDER BY last_name;
last_name
-----
Doe
Key
Lyman

```

Рис. 165. Результат виконання запиту вибору прізвищ під час сортування за прізвищем з таблиці **employee**

Вибрати прізвища незалежно від іншої інформації про співробітників за допомогою команди **DISTINCT** (рис. 166).

```

avocoder=# SELECT DISTINCT last_name FROM employee ORDER BY last_name;
last_name
-----
Doe
Key
Lyman

```

Рис. 166. Результат виконання запиту вибору прізвища незалежно від іншої інформації про співробітників за допомогою команди **DISTINCT** з таблиці **employee**

Виберемо всі записи щодо співробітників жіночої статі – Female (оскільки їх немає, то значень 0) за допомогою команди **WHERE** (рис. 167).

```

avocoder=# SELECT * FROM employee WHERE gender = 'Female';
id | first_name | last_name | gender | email | date_of_birth
---+-----+-----+-----+-----+-----
(0 rows)

```

Рис. 167. Результат виконання запиту вибору всі записи щодо співробітників жіночої статі за допомогою команди **WHERE** з таблиці **employee**

Виберемо всі записи щодо співробітників чоловічої статі – MALE (одне значення) за допомогою команди **AND** (рис. 168).

```

avocoder=# SELECT * FROM employee WHERE gender = 'MALE' AND last_name = 'Lyman';
id | first_name | last_name | gender | email | date_of_birth
---+-----+-----+-----+-----+-----
3 | Konst | Lyman | MALE | dpr@gmail.com | 2015-09-21
(1 row)

```

Рис. 168. Результат виконання запиту вибору всі записи щодо співробітників чоловічої статі за допомогою команди **AND** з таблиці **employee**

Сформуємо запит з використанням **OR** (рис. 169).

```

avecoder=# SELECT * FROM employee WHERE gender = 'MALE' AND (last_name = 'Lymar' OR last_name = 'Key');
 id | first_name | last_name | gender | email | date_of_birth
-----+-----+-----+-----+-----+-----
  2 | Jone      | Key      | MALE  | kre@gmail.com | 2014-09-21
  3 | Konst     | Lymar    | MALE  | dpr@gmail.com | 2015-09-21

```

Рис. 169. Результат виконання запиту з використанням команди **OR** з таблиці **employee**

Сформуємо запит на вибір перших 2-х записів за допомогою команди **LIMIT** (рис. 170).

```

avecoder=# SELECT * FROM employee LIMIT 2;
 id | first_name | last_name | gender | email | date_of_birth
-----+-----+-----+-----+-----+-----
  1 | Jone      | Doe      | MALE  | Jd@gmail.com | 2024-09-21
  2 | Jone      | Key      | MALE  | kre@gmail.com | 2014-09-21

```

Рис. 170. Результат виконання запиту на вибір перших 2-х записів за допомогою команди **LIMIT** з таблиці **employee**

Сформуємо запит на відбір записів, починаючи з 1-ї позиції 2-го запису за допомогою команди **OFFSET** (рис. 171).

```

avecoder=# SELECT * FROM employee OFFSET 1 LIMIT 2;
 id | first_name | last_name | gender | email | date_of_birth
-----+-----+-----+-----+-----+-----
  2 | Jone      | Key      | MALE  | kre@gmail.com | 2014-09-21
  3 | Konst     | Lymar    | MALE  | dpr@gmail.com | 2015-09-21

```

Рис. 171. Результат виконання запиту на вибір записів, починаючи з 1-ї позиції 2-го запису за допомогою команди **OFFSET** з таблиці **employee**

Сформуємо запит на відбір записів, починаючи з 1-ї позиції 2-го запису за допомогою команди **FETCH FIRST** (рис. 172).

```

avecoder=# SELECT * FROM employee OFFSET 1 FETCH FIRST 2 ROW ONLY;
 id | first_name | last_name | gender | email | date_of_birth
-----+-----+-----+-----+-----+-----
  2 | Jone      | Key      | MALE  | kre@gmail.com | 2014-09-21
  3 | Konst     | Lymar    | MALE  | dpr@gmail.com | 2015-09-21

```

Рис. 172. Результат виконання запиту на вибір записів, починаючи з 1-ї позиції 2-го запису за допомогою команди **FETCH FIRST** з таблиці **employee**

Виконання завдання 2

Реалізація команд запитів SQL: IN, BETWEEN, LIKE, iLIKE, GROUP BY, HAVING, ALIAS, COALESCE

Використання команди **IN** при альтернативному виборі під час формування запиту (рис. 173).

```
avocoder=# SELECT * FROM employee WHERE date_of_birth IN ('2014-09-21', '2015-09-21');
```

id	first_name	last_name	gender	email	date_of_birth
2	Jone	Key	MALE	kre@gmail.com	2014-09-21
3	Konst	Lymar	MALE	dpr@gmail.com	2015-09-21

Рис. 173. Результат використання команди **IN** при альтернативному виборі під час формування запиту з таблиці **employee**

Використання команди **BETWEEN** для формування запиту вибору переліку дат, що знаходяться між граничними датами, включно з межами (рис. 174).

```
avocoder=# SELECT * FROM employee
avocoder=# WHERE date_of_birth
avocoder=# BETWEEN '2014-09-21' AND '2024-09-21';
```

id	first_name	last_name	gender	email	date_of_birth
1	Jone	Doe	MALE	Jd@gmail.com	2024-09-21
2	Jone	Key	MALE	kre@gmail.com	2014-09-21
3	Konst	Lymar	MALE	dpr@gmail.com	2015-09-21

Рис. 174. Результат використання команди **BETWEEN** для формування запиту вибору переліку дат, що знаходяться між граничними датами, включно з межами з таблиці **employee**

Використання команди **LIKE** під час формування запиту на вибір електронних адрес співробітників (email) за певним шаблоном адреси (знак % означає, що це може бути різна кількість символів) (рис. 175).

```
avocoder=# SELECT * FROM employee WHERE email LIKE '%.com';
```

id	first_name	last_name	gender	email	date_of_birth
1	Jone	Doe	MALE	Jd@gmail.com	2024-09-21
2	Jone	Key	MALE	kre@gmail.com	2014-09-21
3	Konst	Lymar	MALE	dpr@gmail.com	2015-09-21

Рис. 175. Результат використання команди **LIKE** під час формування запиту на вибір електронних адрес співробітників (email) за певним шаблоном адреси (знак % означає, що це може бути різна кількість символів) з таблиці **employee**

Необхідно вибрати всіх співробітників, імена яких починаються з літери «J» за допомогою застосування команди **LIKE** (рис. 176).

```
avecoder=# SELECT * FROM employee WHERE first_name LIKE 'J%';
```

id	first_name	last_name	gender	email	date_of_birth
1	Jone	Doe	MALE	Jd@gmail.com	2024-09-21
2	Jone	Key	MALE	kre@gmail.com	2014-09-21

Рис. 176. Результат обрання всіх співробітників, імена яких починаються з літери «J» за допомогою застосування команди **LIKE** з таблиці **employee**

Необхідно визначити кількість усіх співробітників, імена яких однакові за допомогою команд **COUNT** і **GROUP BY** (рис. 177).

```
avecoder=# SELECT first_name, COUNT(*) FROM employee GROUP BY first_name;
```

first_name	count
Konst	1
Jone	2

Рис. 177. Результат визначення кількості усіх співробітників, імена яких однакові за допомогою команд **COUNT** і **GROUP BY** з таблиці **employee**

Необхідно визначити кількість усіх співробітників, імена яких однакові за допомогою команд **COUNT**, **GROUP BY** і **HAVING** (рис. 178).

```
avecoder=# SELECT first_name, COUNT(*) FROM employee GROUP BY first_name HAVING COUNT(*) > 1;
```

first_name	count
Jone	2

Рис. 178. Результат визначення кількості усіх співробітників, імена яких однакові за допомогою команд **COUNT**, **GROUP BY** і **HAVING** з таблиці **employee**

Необхідно визначити кількість усіх співробітників, імена яких однакові в спадному за алфавітом порядку за допомогою команд **COUNT**, **GROUP BY** і **HAVING** (рис. 179).

```
avecoder=# SELECT first_name, COUNT(*) FROM employee GROUP BY first_name HAVING COUNT(*) > 1;
avecoder=# ORDER BY first_name DESC;
```

first_name	count
Jone	2

Рис. 179. Результат визначення кількості усіх співробітників, імена яких однакові в спадному за алфавітом порядку за допомогою команд **COUNT**, **GROUP BY** і **HAVING** з таблиці **employee**

Для того, щоб не повертати порожні поля, а заповнити в разі відсутності інформації в них, у запиті необхідно використати команду **COALESCE** (у цьому випадку в полі email немає незаповнених значень) (рис. 180).

```
avocoder=# SELECT COALESCE(email, 'not applicable') FROM employee;
coalesce
-----
Jd@gmail.com
kre@gmail.com
dpr@gmail.com
```

Рис. 180. Результат використання команди **COALESCE** з таблиці **employee**

Для присвоювання сутностям тимчасових імен з використанням команди **AS** (рис. 181).

```
avocoder=# SELECT id, first_name AS name, last_name AS surname, gender AS sex, email, date_of_birth
avocoder=# FROM employee;
id | name | surname | sex | email | date_of_birth
---+---+---+---+---+---
1 | Jone | Doe | MALE | Jd@gmail.com | 2024-09-21
2 | Jone | Key | MALE | kre@gmail.com | 2014-09-21
3 | Konst | Lymar | MALE | dpr@gmail.com | 2015-09-21
```

Рис. 181. Результат присвоювання сутностям тимчасових імен з використанням команди **AS** з таблиці **employee**

Дата і час

Запит на виведення поточних дати і часу системи (рис. 182).

```
avocoder=# SELECT NOW();
now
-----
2024-09-24 23:35:59.532453+03
```

Рис. 182. Результат виконання запиту на виведення поточних дати і часу системи за допомогою функції маніпулювання з часом **NOW**

Відфільтрувати тільки за датою (рис. 183).

```
avocoder=# SELECT NOW()::DATE;
now
-----
2024-09-24
```

Рис. 183. Результат виконання запиту щодо відфільтрування тільки за датою за допомогою функції маніпулювання з часом **DATE**

Відфільтрувати тільки за часом (рис. 184).

```

avocoder=# SELECT NOW()::TIME;
          now
-----
23:40:14.359161

```

Рис. 184. Результат виконання запиту щодо відфільтрування тільки за часом за допомогою функції маніпулювання з часом **TIME**

Функція маніпуляції з часом **INTERVAL**

Віднімаємо від поточної дати 1 рік (рис. 185).

```

avocoder=# SELECT NOW() - INTERVAL '1 YEAR';
          ?column?
-----
2023-09-24 23:44:56.000348+03

```

Рис. 185. Результат виконання запиту щодо віднімання від поточної дати 1 рік за допомогою функцію маніпулювання з часом **INTERVAL**

Відняти від поточної дати 100 років (рис. 186).

```

avocoder=# SELECT NOW() - INTERVAL '100 YEAR';
          ?column?
-----
1924-09-24 23:46:37.879074+02

```

Рис. 186. Результат виконання запиту щодо віднімання від поточної дати 100 років

Відняти від поточної дати 10 місяців (рис. 1873).

```

avocoder=# SELECT NOW() - INTERVAL '10 MONTH';
          ?column?
-----
2023-11-24 23:48:02.303299+02

```

Рис. 187. Результат виконання запиту щодо віднімання від поточної дати 10 місяців

Відняти від поточної дати 10 днів (рис. 188).

```

avocoder=# SELECT NOW() - INTERVAL '10 DAYS';
          ?column?
-----
2024-09-14 23:52:19.385044+03

```

Рис. 188. Результат виконання запиту щодо віднімання від поточної дати 10 днів

Додати до поточної дати 100 років (рис. 189).

```
avocoder=# SELECT NOW() + INTERVAL '100 YEAR';
?column?
-----
2124-09-24 23:53:21.479885+03
```

Рис. 189. Результат виконання запиту щодо додавання до поточної дати 100 років

Витяг із дати року (рис. 190).

```
avocoder=# SELECT EXTRACT(YEAR FROM NOW());
extract
-----
2024
```

Рис. 190. Результат виконання запиту щодо витягу із дати року

Витяг із дати місяця (рис. 191).

```
avocoder=# SELECT EXTRACT(MONTH FROM NOW());
extract
-----
9
```

Рис. 191. Результат виконання запиту щодо витягу із дати місяця

Витяг із дати дня тижня (рис. 192).

```
avocoder=# SELECT EXTRACT(DOW FROM NOW());
extract
-----
3
```

Рис. 192. Результат виконання запиту щодо витягу із дати тижня

Дізнатися вік усіх співробітників (рис. 193).

```
avocoder=# SELECT first_name, last_name, gender, email, date_of_birth, AGE(NOW(), date_of_birth)
avocoder=# AS age FROM employee;
first_name | last_name | gender | email      | date_of_birth | age
-----+-----+-----+-----+-----+-----
Jones      | Doe       | MALE   | Jd@gmail.com | 2024-09-21    | 4 days 00:08:31.521318
Jones      | Key       | MALE   | kre@gmail.com | 2014-09-21    | 10 years 4 days 00:08:31.521318
Konst      | Lyman     | MALE   | dpr@gmail.com | 2015-09-21    | 9 years 4 days 00:08:31.521318
```

Рис. 193. Результат виконання запиту щоб дізнатися вік усіх співробітників

Виконання завдання 3

Реалізація обмеження та перевірки – UNIQUE, CHECK

Дізнатися чи є дублікати у наявних записів у таблиці (рис. 194).

```
avocoder=# SELECT email, count(*) FROM employee GROUP BY email;
```

email	count
dpr@gmail.com	2
kre@gmail.com	1

Рис. 194. Результат виконання запиту щодо наявності дублікатів записів у таблиці

Поля, які повторюються і порожні (рис. 195).

```
avocoder=# SELECT email, count(*) FROM employee GROUP BY email HAVING COUNT(*) > 1;
```

email	count
	2

Рис. 195. Результат виконання запиту щодо повторюваності полів і порожніх полів

Вкажемо на унікальність адреси в email (рис. 196).

```
avocoder=# ALTER TABLE employee ADD CONSTRAINT unique_email_address UNIQUE (email);
ERROR: could not create unique index "unique_email_address"
ПОДРОБНОСТИ: Key (email)=() is duplicated.
```

```
avocoder=# ALTER TABLE employee ADD CONSTRAINT unique_email_address UNIQUE (email);
ALTER TABLE
avocoder=# \d employee
```

Имя	Тип	Параметры	is nullable	Примечания
id	bigint		not null	nextval('employee_id_seq'::regclass)
first_name	character varying(50)		not null	
last_name	character varying(50)		not null	
gender	character varying(6)		not null	
email	character varying(150)		not null	
date_of_birth	date		not null	

```
Индиксы:
"employee_pkey" PRIMARY KEY, btree (id)
"unique_email_address" UNIQUE CONSTRAINT, btree (email)
```

Рис. 196. Результат виконання запиту щодо унікальності адреси в email

Додамо співробітника, який має стать panda (рис. 197).

```

avecoder=# INSERT INTO employee (
avecoder(# first_name, last_name, gender, email, date_of_birth)
avecoder=# VALUES ('Arkada', 'Monki', 'panda', 'dhs@khpi.edu.ua', '1997-09-30');
INSERT 0 1

```

```

avecoder=# SELECT * FROM employee;
 id | first_name | last_name | gender | email | date_of_birth
-----+-----+-----+-----+-----+-----
  2 | Jones      | Key       | MALE   | kre@gmail.com | 2014-09-21
  3 | Konst     | Lyman     | MALE   | dpr@gmail.com | 2015-09-21
  1 | Johan     | Krif      | MALE   |             | 2000-01-31
  6 | Arkada    | Monki     | panda  | dhs@khpi.edu.ua | 1997-09-30

```

```

avecoder=# SELECT DISTINCT gender FROM employee;
gender
-----
panda
MALE

```

Рис. 197. Результат виконання запиту щоб дізнатися вік усіх співробітників

Додамо представника жіночої статі (рис. 198).

```

avecoder=# INSERT INTO employee (
avecoder(# first_name, last_name, gender, email, date_of_birth)
avecoder=# VALUES ('Tatiana', 'Montian', 'FEMALE', 'lop@khpi.edu.ua', '1973-05-03');
INSERT 0 1
avecoder=# SELECT * FROM employee;
 id | first_name | last_name | gender | email | date_of_birth
-----+-----+-----+-----+-----+-----
  2 | Jones      | Key       | MALE   | kre@gmail.com | 2014-09-21
  3 | Konst     | Lyman     | MALE   | dpr@gmail.com | 2015-09-21
  1 | Johan     | Krif      | MALE   |             | 2000-01-31
  6 | Arkada    | Monki     | panda  | dhs@khpi.edu.ua | 1997-09-30
  7 | Tatiana   | Montian   | FEMALE  | lop@khpi.edu.ua | 1973-05-03

```

Рис. 198. Результат виконання запиту щоб дізнатися вік усіх співробітників

Додамо обмеження, яке здійснюватиме перевірку (рис. 199).

```

avecoder=# ALTER TABLE employee ADD CONSTRAINT gender_constraint CHECK (gender = 'FEMALE' OR gender = 'MALE');
ERROR: check constraint "gender_constraint" of relation "employee" is violated by some row
avecoder=#

```

Рис. 199. Результат виконання запиту щоб дізнатися вік усіх співробітників

Видає помилку, оскільки ще є одна стаття, яка не врахована в умові. Для цього видалимо статтю 'panda' (рис. 200).

```

avecoder=# DELETE FROM employee WHERE gender = 'panda';
DELETE 1

```

Рис. 200. Результат виконання запиту щоб дізнатися вік усіх співробітників

Після успішного видалення, додамо обмеження (рис. 201).

```

avecoder=# ALTER TABLE employee ADD CONSTRAINT gender_constraint CHECK (gender = 'FEMALE' OR gender = 'MALE');
ALTER TABLE

```

Рис. 201. Результат виконання запиту щоб дізнатися вік усіх співробітників

Спробуємо знову вставити Arkada (рис. 202).

```
avecoder=# INSERT INTO employee (  
avecoder(# first_name, last_name, gender, email, date_of_birth)  
avecoder=# VALUES ('Arkada', 'Monki', 'panda', 'dhs@khpi.edu.ua', '1997-09-30');  
ERROR: new row for relation "employee" violates check constraint "gender_constraint"  
ПОДРОБНОСТИ: Failing row contains (8, Arkada, Monki, panda, dhs@khpi.edu.ua, 1997-09-30).
```

Рис. 202. Результат виконання запиту щоб дізнатися вік усіх співробітників

Але видається повідомлення про помилку, оскільки набуло чинності зазначене обмеження.

Контрольні запитання та завдання

1. Яке призначення та структура команд SELECT, ORDER BY, DISTINCT, WHERE, AND, OR, LIMIT, OFFSET, FETCH?
2. Яке призначення та структура команд IN, BETWEEN, LIKE, iLIKE, GROUP BY, HAVING, ALIAS, COALESCE?
3. Яке призначення та структура команд UNIQUE, CHECK?

ВСТАНОВЛЕННЯ ПЕРВИННИХ КЛЮЧІВ ЗА ДОПОМОГОЮ SQL SHELL (КОНСОЛІ)

Meta:

Придбати навички із встановлення первинних ключів в базі даних засобами СУБД PostgreSQL

Завдання:

1. Встановлення первинного ключа в таблицю бази даних із вже існуючим id та .
2. виправлення помилок щодо встановлення первинного ключа.

Виконання завдання 1

Вставка співробітника в таблицю з уже існуючим id.

У результаті видається повідомлення про помилку (рис. 203).

```
avecoder=# INSERT INTO employee (  
avecoder(# id, first_name, last_name, gender, email, date_of_birth)  
avecoder-# VALUES('1', 'Johan', 'Krif', 'MALE', '', '2000-01-31');  
ERROR:  duplicate key value violates unique constraint "employee_pkey"  
ПОДРОБНОСТИ:  Key (id)=(1) already exists.
```

Рис. 203. Результат додавання співробітника в таблицю з уже існуючим id

У цьому випадку первинний ключ виступає обмежувачем. Скинемо обмежувач:

```
avocoder=# ALTER TABLE employee DROP CONSTRAINT employee_pkey;
```

Обмежувач було видалено – `ALTER TABLE`. Перевіримо таблицю (рис. 204).

```
avacoder=# \d employee
```

	тип	присущ "public.employee"	по умолчанию NULL	ссылка на класс
id	bigint		not null	nextval('employee_id_seq'::regclass)
first_name	character varying(50)		not null	
last_name	character varying(50)		not null	
gender	character varying(6)		not null	
email	character varying(150)			
date_of_birth	date		not null	

Рис. 204. Результат перевірки таблиці із врахуванням видалення обмежувача

Умова, що **employee_pkey** (первинний ключ) відсутній, отже, відсутній обмежувач.

Ще раз вставимо співробітника в таблицю з уже існуючим id:

```
avecoder=# INSERT INTO employee (
avecoder(# id, first_name, last_name, gender, email, date_of_birth)
avecoder=# VALUES('1', 'Johan', 'Krif', 'MALE', '', '2000-01-31');
INSERT 0 1
```

Таким чином вставка спрацювала (рис. 205).

```
avecoder=# SELECT * FROM employee WHERE id =1;
 id | first_name | last_name | gender | email | date_of_birth
-----+-----+-----+-----+-----+-----
  1 | Jone       | Doe       | MALE   | Jd@gmail.com | 2024-09-21
  1 | Johan      | Krif      | MALE   |              | 2000-01-31
```

Рис. 205. Результат виконання вставки без врахування обмежувача на записи з однаковими id

Таким чином з'явилися два записи з однаковим id.

Виконання завдання 2

Повернемо первинний ключ. У результаті видається помилка (рис. 206).

```
avecoder=# ALTER TABLE employee ADD PRIMARY KEY(id);
ERROR:  could not create unique index "employee_pkey"
ПОДРОБНОСТИ:  Key (id)=(1) is duplicated.
```

Рис. 206. Результат повернення обмеження щодо встановлення первинного ключа

Якщо сформулювати запит на видалення запису з id=1, то будуть видалені відразу два записи. Якщо сформуємо таблицю, то побачимо, що відсутні дані запису (рис. 207).

```
avecoder=# DELETE FROM employee WHERE id = 1;
DELETE 2
avecoder=# SELECT * FROM employee WHERE id =1;
 id | first_name | last_name | gender | email | date_of_birth
-----+-----+-----+-----+-----+-----

```

Рис. 207. Результат виконання запиту на видалення запису з id=1

Додавання первинного ключа в таблицю (рис. 208).

```

avecoder=# INSERT INTO employee (
avecoder=# first_name, last_name, gender, email, date_of_birth)
avecoder=# VALUES('Johan', 'Krif', 'MALE', '', '2000-01-31');
INSERT 0 1
avecoder=# \d employee

```

Таблиця	Тип	Параметри	Nullable	Секвенція
id	bigint		not null	nextval('employee_id_seq'::regclass)
first_name	character varying(50)		not null	
last_name	character varying(50)		not null	
gender	character varying(6)		not null	
email	character varying(150)		not null	
date_of_birth	date		not null	

```

avecoder=# ALTER TABLE employee ADD PRIMARY KEY(id);
ALTER TABLE
avecoder=# \d employee

```

Таблиця	Тип	Параметри	Nullable	Секвенція
id	bigint		not null	nextval('employee_id_seq'::regclass)
first_name	character varying(50)		not null	
last_name	character varying(50)		not null	
gender	character varying(6)		not null	
email	character varying(150)		not null	
date_of_birth	date		not null	

```

\efhv:
"employee_pkey" PRIMARY KEY, btree (id)
avecoder=# INSERT INTO employee (
avecoder=# id, first_name, last_name, gender, email, date_of_birth)
avecoder=# VALUES('0001', 'Johan', 'Krif', 'MALE', '', '2000-01-31');
INSERT 0 1
avecoder=# \d employee

```

Таблиця	Тип	Параметри	Nullable	Секвенція
id	bigint		not null	nextval('employee_id_seq'::regclass)
first_name	character varying(50)		not null	
last_name	character varying(50)		not null	
gender	character varying(6)		not null	
email	character varying(150)		not null	
date_of_birth	date		not null	

```

\efhv:
"employee_pkey" PRIMARY KEY, btree (id)
avecoder=# SELECT * FROM employee;

```

id	first_name	last_name	gender	email	date_of_birth
2	Jone	Key	MALE	kre@gmail.com	2014-09-21
3	Konst	Lymar	MALE	dpr@gmail.com	2015-09-21
4	Jone	Doe	MALE		2000-01-01
5	Johan	Krif	MALE		2000-01-31
1	Johan	Krif	MALE		2000-01-31

Рис. 208. Результат додавання первинного ключа в таблицю

Видалимо запис із id=5 та id=4 (рис. 209).

```

avecoder=# DELETE FROM employee WHERE id = 5;
DELETE 1
avecoder=# DELETE FROM employee WHERE id = 4;
DELETE 1

```

Рис. 209. Інструкції та повідомлення про їх виконання

Контрольні запитання та завдання

1. Яку роль виконує первинний ключ в таблицях бази даних?
2. Які наслідки викликає видалення обмеження щодо первинного ключа?

ЛАБОРАТОРНА РОБОТА № 8

ВСТАНОВЛЕННЯ ЗОВНІШНІХ КЛЮЧІВ ЗАСОБАМИ MYSQL TA POSTGRESQL

Мета:

Отримати навички із призначення зовнішніх ключів із використанням СУБД MySQL та СУБД PostgreSQL

Завдання:

1. Створити зовнішні ключі за допомогою MySQL.
2. Створити зовнішні ключі за допомогою PostgreSQL.

Виконання завдання 1

Раніше було розглянуто поняття первинного ключа і зовнішнього ключа, які утворюють зв'язок між таблицями. Так, ключ **catalogs.id_catalog** є первинним ключем таблиці **catalogs**, а ключ **products.id_catalog** – зовнішній ключ таблиці **products**. Сукупність ключів **catalogs.id_catalog** та **products.id_catalog** утворює зв'язок між таблицями **catalogs** и **products**.

Вміст таблиці **catalogs**:

```
mysql> SELECT * FROM catalogs;
```

id_catalog	name
1	Процесори
2	Материнські плати
3	Відеоадаптери
4	Жорсткі диски
5	Оперативна пам'ять

Нехай у зв'язку з реорганізацією електронного магазину потрібно видалити запис каталогу «Жорсткі диски». Для цього можна вдатися до оператора **DELETE**:

```
mysql> DELETE FROM catalogs WHERE name = 'Жорсткі диски';
```

```
mysql> SELECT * FROM catalogs;
```

id_catalog	name
------------	------

1		Процесори	
2		Материнські плати	
3		Відеоадаптери	
5		Оперативна пам'ять	
+-----+-----+-----+-----+			

Після видалення розділу з первинним ключем **id_catalog**, рівним 4, у таблиці **products** залишилися товарні позиції, які належать віддаленому розділу:

```
mysql> SELECT id_product, id_catalog, name FROM products WHERE
id_catalog = 4;
```

+-----+-----+-----+-----+			
id_product		id_catalog	name
+-----+-----+-----+-----+			
20		4	Maxtor 6Y120P0
21		4	Maxtor 6B200P0
22		4	Samsung SP0812C
23		4	Seagate Barracuda ST3160023A
24		4	Seagate ST3120026A
+-----+-----+-----+-----+			

Таку ситуацію називають порушенням посилальної цілісності. У межах реляційної моделі таблицю **catalogs** називають предком, а **products** – нащадком. Загалом існує чотири типи змін бази даних, які можуть порушити посилальну цілісність.

Додавання нового рядка-нащадка.

Додавання в електронний магазин товарної позиції, що не належить жодному з елементів каталогу, тобто якщо поле **id_catalog** таблиці **products** приймає, наприклад, значення **104**. Слід зазначити, що додавання в таблицю **catalogs** рядка, для якого немає відповідності в таблиці **products**, не викликає суперечностей, оскільки існування каталогу без товарних позицій цілком допустиме явище.

Оновлення зовнішнього ключа в рядку-нащадку.

Це та сама проблема, що й у попередній ситуації, але виражена в іншій формі. Під час оновлення поля **id_catalog** таблиці **products**, тобто під час перенесення товарної позиції від одного елемента каталогу до іншого необхідно, щоб каталог із новим значенням **id_catalog** існував у таблиці **catalogs**, інакше товарна позиція знову повисає в повітрі.

Видалення рядка-предка.

Цю ситуацію описано раніше вище. Видалення елемента каталогу з таблиці **catalogs** призводить до того, що низка товарних позицій належать до неіснуючої позиції каталогу.

Оновлення первинного ключа в рядку-предку.

Це інша форма проблеми, розглянутої в попередньому пункті. Під час оновлення первинного ключа каталогу в товарних позицій, які належать до цього каталогу, зовнішній ключ не змінюється.

Засоби підтримки посилювальної цілісності мови SQL дають змогу обробляти кожен з чотирьох описуваних ситуацій.

Перша проблема (додавання нового рядка-нащадка) розв'язується шляхом перевірки значень у стовпцях зовнішнього ключа перед виконанням оператора **insert**. Якщо передані оператору **insert** значення не дорівнюють жодному зі значень первинного ключа, то повертається помилка додавання запису.

Друга проблема (оновлення таблиці-нащадка) вирішується аналогічно шляхом перевірки нового значення зовнішнього ключа. Якщо такого значення серед первинних ключів таблиці-предка не виявлено, повертається помилка.

Третя проблема (видалення рядка-предка) може потребувати різної реакції залежно від контексту і предметної області. Так, стосовно електронного магазину під час видалення запису каталогу з таблиці **catalogs** можуть бути застосовані такі дії:

- не видаляти елемент каталогу доти, доки товарні позиції з таблиці, які йому належать, не зникнуть **products** не будуть співвіднесені з іншим записом каталогу;
- автоматично видалити всі товарні позиції таблиці **products**, що належать елементу каталогу, який видаляється;
- для товарних позицій, що належать елементу каталогу, який видаляється, у полі **id_catalog** помістити значення **NULL**, яке сигналізуватиме про те, що елемент каталогу для цих товарних позицій не відомий;
- для товарних позицій, що належать запису каталогу, який видаляється, у полі **id_catalog** помістити значення за замовчуванням, відмінне від **null**, наприклад, значення **id_catalog** для елемента каталогу «Різне».

Точно такі самі проблеми виникають у четвертій ситуації (оновлення первинного ключа в таблиці-предку). Для завдання всіх цих дій у стандарті SQL передбачено спеціальні ключові слова. До цього моменту при визначенні структури таблиці явним чином позначався тільки первинний ключ. Однак для того щоб задати правила видалення й оновлення, потрібна також явна вказівка вторинного ключа, що задається конструкцією **FOREIGN KEY**, яка має такий синтаксис:

```
FOREIGN KEY [name_key] (col1, ...) REFERENCES tbl (tbl_col, ...)
[ON DELETE {CASCADE|SET NULL|NO ACTION|RESTRICT|SET DEFAULT}]
[ON UPDATE {CASCADE |SET NULL|NO ACTION|RESTRICT|SET DEFAULT}]
```

Ця конструкція дає змогу задати зовнішній ключ із необов'язковим ім'ям **name_key** на стовпчиках, що перелічуються в круглих дужках (один або кілька). Ключове слово **references** вказує таблицю *tbl*, на яку посилається зовнішній ключ, у круглих дужках задаються імена стовпців. Необов'язкові конструкції **ON DELETE** та **ON UPDATE** дають змогу визначити поведінку СУБД під час видалення та оновлення рядків із таблиці-предка, відповідно. Параметри, що йдуть за цими ключовими словами, мають такі значення:

CASCADE – у разі видалення або оновлення запису, що містить первинний ключ, записи з посиланнями на це значення в таблиці-нащадку видаляються або оновлюються автоматично;

SET NULL – у разі видалення або оновлення запису, що містить первинний ключ, у таблиці-нащадку значення зовнішнього ключа, що посилається на таблицю-предка, встановлюються в **NULL**;

NO ACTION – при видаленні або оновленні записів, що містять первинний ключ, з таблицею-нащадком ніяких додаткових дій не проводиться;

RESTRICT – якщо в таблиці-нащадку є записи, що посилаються на первинний ключ таблиці-предка, при видаленні або оновленні записів з таким первинним ключем повертається помилка. СУБД не дозволяє змінювати або видаляти запис із первинним ключем, поки не залишиться жодного посилання з таблиці-нащадка;

SET DEFAULT – згідно зі стандартом SQL, під час видалення або оновлення первинного ключа в таблиці-нащадку для записів, що посилаються на нього, у полі зовнішнього ключа має встановлюватися

значення за замовчуванням. Однак у СУБД MySQL це ключове слово зарезервовано, але не обробляється.

Забезпечимо таблиці навчальної бази даних shop зовнішніми ключами. Слід пам'ятати, що зовнішні ключі підтримуються тільки таблицями типу InnoDB, тому всі таблиці мають бути перетворені до цього типу. Це можна здійснити або за допомогою оператора **ALTER TABLE**, або створивши таблиці заново з використанням оператора **CREATE TABLE**:

```
mysql>CREATE TABLE catalogs (  
->id_catalog int(11) NOT NULL auto_increment,  
->name tinytext NOT NULL,  
->PRIMARY KEY (id_catalog))  
->ENGINE=InnoDB DEFAULT CHARSET=cp1251;
```

```
mysql>CREATE TABLE products (  
->id_product int(11) NOT NULL auto_increment,  
->name tinytext NOT NULL,  
->price decimal(7,2) default '0.00',  
->count int(11) default '0',  
->mark float(4,1) NOT NULL default '0.0',  
->description text,  
->id_catalog int(11) NOT NULL default '0',  
->PRIMARY KEY (id_product),  
->FOREIGN KEY (id_catalog) REFERENCES catalogs(id_catalog)  
->ON DELETE CASCADE  
->ON UPDATE CASCADE  
->) ENGINE=InnoDB DEFAULT CHARSET=cp1251;
```

Вище в табличному просторі InnoDB створюються таблиці **catalogs** та **products**. В таблиці **products** поле **id_catalog** оголошено як зовнішній ключ (**FOREIGN KEY**) з правилом видалення та оновлення **CASCADE**. Це означає, що оновлення таблиці **catalogs** призводить до автоматичного оновлення таблиці **products**:

```
mysql> UPDATE catalogs SET id_catalog = 10 WHERE id_catalog = 4;  
mysql> SELECT id_product, id_catalog, name FROM products;
```

```
+-----+-----+-----+  
| id_product | id_catalog | name          |  
+-----+-----+-----+
```

1	1	Celeron 1.8	
2	1	Celeron 2.0GHz	
3	1	Celeron 2.4GHz	
4	1	Celeron D 320 2.4GHz	
5	1	Celeron D 325 2.53GHz	
6	1	Celeron D 315 2.26GHz	
7	1	Intel Pentium 4 3.2GHz	
8	1	Intel Pentium 4 3.0GHz	
9	1	Intel Pentium 4 3.0GHz	
10	2	Gigabyte GA-8I848P-RS	
11	2	Gigabyte GA-8IG1000	
12	2	Gigabyte GA-8IPE1000G	
13	2	Asustek P4C800-E Delux	
14	2	Asustek P4P800-VM\ i865G	
15	2	Epoх EP-4PDA3I	
16	3	ASUSTEK A9600XT/TD	
17	3	ASUSTEK V9520X	
18	3	SAPPHIRE 256MB RADEON 9550	
19	3	GIGABYTE AGP GV-N59X128D	
20	10	Maxtor 6Y120P0	
21	10	Maxtor 6B200P0	
22	10	Samsung SP0812C	
23	10	Seagate Barracuda ST3160023A	
24	10	Seagate ST3120026A	
25	5	DDR-400 256MB Kingston	
26	5	DDR-400 256MB Hynix Original	
27	5	DDR-400 256MB PQI	
28	5	DDR-400 512MB Kingston	
29	5	DDR-400 512MB PQI	
30	5	DDR-400 512MB Hynix	

+-----+-----+-----+

А видалення будь-якого з елементів каталогу в таблиці **catalogs** призведе до того, що всі записи таблиці **products**, поле **id_catalog** яких дорівнює первинному ключу запису каталогу, що видаляється, також буде видалено з таблиці:

```
mysql> DELETE FROM catalogs WHERE id_catalog > 2;
mysql> SELECT * FROM catalogs;
```

id_catalog	name
1	Процесори
2	Материнські плати

```
mysql> SELECT id_product, id_catalog, name FROM products;
```

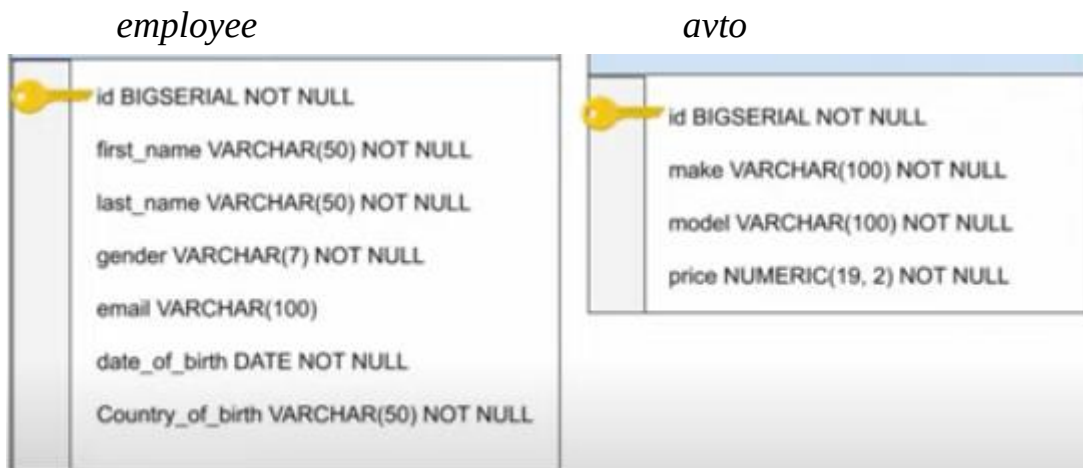
id_product	id_catalog	name
1	1	Celeron 1.8
2	1	Celeron 2.0GHz
3	1	Celeron 2.4GHz
4	1	Celeron D 320 2.4GHz
5	1	Celeron D 325 2.53GHz
6	1	Celeron D 315 2.26GHz
7	1	Intel Pentium 4 3.2GHz
8	1	Intel Pentium 4 3.0GHz
9	1	Intel Pentium 4 3.0GHz
10	2	Gigabyte GA-8I848P-RS
11	2	Gigabyte GA-8IG1000
12	2	Gigabyte GA-8IPE1000G
13	2	Asustek P4C800-E Delux
14	2	Asustek P4P800-VM i865G
15	2	Epoх EP-4PDA3I

Як видно з результату, видалення записів із первинним ключем призводить до автоматичного видалення записів із зовнішнім ключем, що посиляються на нього. Записи із зовнішніми ключами самі можуть виступати як посилання для зовнішніх ключів інших таблиць, і один-єдиний запит може здійснювати досить складні перетворення бази даних. Видалення та оновлення бази даних, коли таблиці пов'язані зовнішнім ключем із правилом **CASCADE**, називається каскадним видаленням або оновленням.

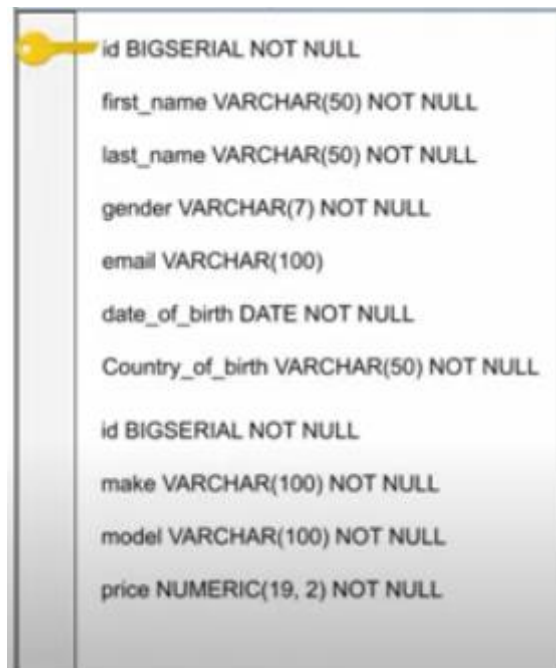
Виконання завдання 2

Компанія кожному співробітнику надає автомобіль, щоб вони змогли дістатися до роботи, але є 3 умови:

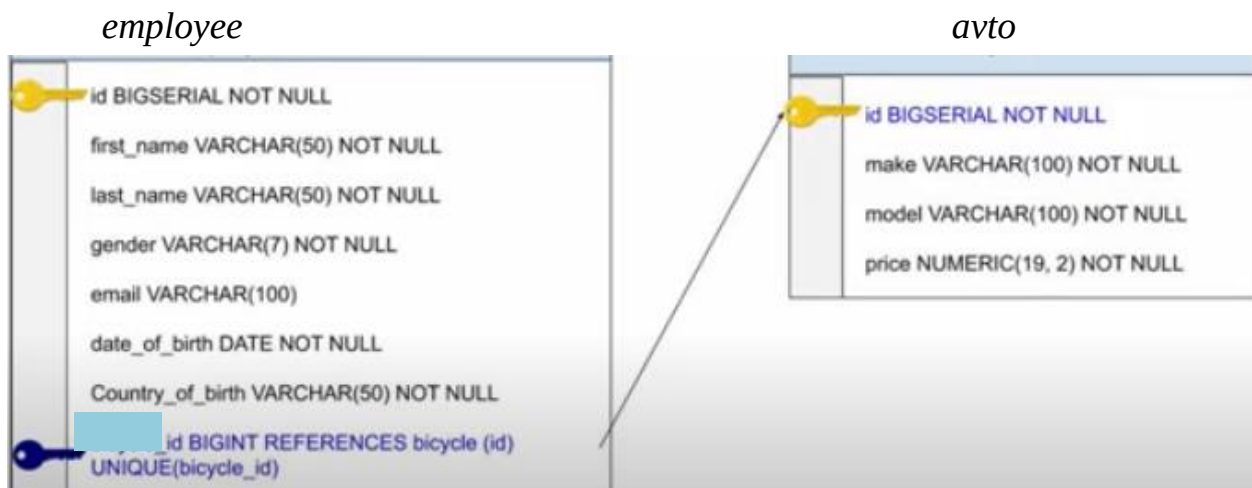
- 1) один працівник – один автомобіль;
 - 2) 1 автомобіль може належати тільки одному співробітнику і передавати іншим не можна;
 - 3) працівник може оформити собі авто, а може і не оформляти.
- Є дві таблиці – працівники (*employee*) і автомобілі (*avto*):



Перший підхід – створити одну таблицю з усіма полями:



У деяких полів є значення обов'язкові до заповнення, тобто **NOT NULL** і це може порушувати умову 3). Тому треба розділити цю таблицю на дві, але при цьому таблиця працівників буде посилатися на таблицю авто за потреби. При цьому знадобляться зовнішні ключі, які є посиланням однієї таблиці до первинного ключа іншої таблиці:



Створимо таблицю *avto*, в якій:

- make – виробник автомобіля;
- type – тип автомобіля;
- price – ціна автомобіля (рис. 210).

```
avocoder=# CREATE TABLE avto (
avocoder=# id BIGSERIAL NOT NULL PRIMARY KEY,
avocoder=# make VARCHAR(100),
avocoder=# type VARCHAR(100) NOT NULL,
avocoder=# price NUMERIC(19, 2) NOT NULL
avocoder=# );
CREATE TABLE
```

Рис. 210. Створення таблиці *avto*

Перевіримо, що створена таблиця присутня в базі даних (рис. 211).

```
avocoder=# \d
```

schema	table	columns	types
public	avto	id	bigint
public	avto_id_seq	id	bigint
public	employee	id, first_name, last_name, gender, email, date_of_birth, Country_of_birth, bicycle_id	bigint, varchar, varchar, varchar, date, varchar, bigint
public	employee_id_seq	id	bigint
public	sweets_types	id, name, price	bigint, varchar, numeric

Рис. 211. Перевірка наявності створеної таблиці *avto* в базі даних

Позначимо зовнішній ключ у таблиці *employee* (рис. 212).

```

avocoder=# \d employee
      table "public.employee"
      id bigint not null nextval('employee_id_seq')
      first_name character varying(50) not null
      last_name character varying(50) not null
      gender character varying(6) not null
      email character varying(150)
      date_of_birth date not null
      "employee_pkey" PRIMARY KEY, btree (id)
      "unique_email_address" UNIQUE CONSTRAINT, btree (email)
      "gender_constraint" CHECK (gender::text = 'FEMALE'::text OR gender::text = 'MALE'::text)

```

Рис. 212. Позначення зовнішнього ключа

Додавання обмеження за допомогою **ALTER**

Зазначити зовнішній ключ **avto_id** типу **BEGINT**, який посилається (**REFERENCES**) на таблицю (**avto**) поля (**id**) (рис. 213).

```

avocoder=# ALTER TABLE employee ADD avto_id BIGINT REFERENCES avto (id);
ALTER TABLE

avocoder=# \d employee
      table "public.employee"
      id bigint not null nextval('employee_id_seq'::regclass)
      first_name character varying(50) not null
      last_name character varying(50) not null
      gender character varying(6) not null
      email character varying(150)
      date_of_birth date not null
      avto_id bigint
      "employee_pkey" PRIMARY KEY, btree (id)
      "unique_email_address" UNIQUE CONSTRAINT, btree (email)
      "gender_constraint" CHECK (gender::text = 'FEMALE'::text OR gender::text = 'MALE'::text)
      "employee_avto_id_fkey" FOREIGN KEY (avto_id) REFERENCES avto(id)

```

Рис. 213. Зазначення зовнішнього ключа **avto_id**

Останнє поле – це **avto_id**, яке є зовнішнім ключем, що посилається на первинний ключ із таблиці **avto**.

Далі виконуємо умову 2) за допомогою обмежувача (рис. 214).

```

avocoder=# ALTER TABLE employee ADD UNIQUE (avto_id);
ALTER TABLE

```

Рис. 214. Виконання умови 2)

Тепер мають бути унікальними як зовнішній ключ **avto_id**, так й **email** (рис. 215).

```

avocoder=# \d employee
      table "public.employee"
      column | data type | character set | collation | storage |
-----+-----+-----+-----+-----+
id           | bigint   |                |           |         |
first_name   | character varying(50) |                |           |         |
last_name    | character varying(50) |                |           |         |
gender       | character varying(6)  |                |           |         |
email        | character varying(150)|                |           |         |
date_of_birth | date      |                |           |         |
avto_id      | bigint   |                |           |         |
Indexes:
    "employee_pkey" PRIMARY KEY, btree (id)
    "employee_avto_id_key" UNIQUE CONSTRAINT, btree (avto_id)
    "unique_email_address" UNIQUE CONSTRAINT, btree (email)
Check constraints:
    "gender_constraint" CHECK (gender::text = 'FEMALE'::text OR gender::text = 'MALE'::text)
Foreign key constraints:
    "employee_avto_id_fkey" FOREIGN KEY (avto_id) REFERENCES avto(id)

```

Рис. 215. Перевірка встановлених обмежень

Ключ **avto_id** присутній як зовнішній ключ у «**employee_avto_id_fkey**» **FOREIGN KEY (avto_id) REFERENCES avto(id)**, так і як обмеження «**employee_avto_id_key**» **UNIQUE CONSTRAINT, btree (avto_id)**.

Заповнити таблицю *avto* автомобілями (рис. 216).

```

avocoder=# insert into avto (make, type, price) VALUES ('Reno', 'Daster', '25000.00');
INSERT 0 1
avocoder=# insert into avto (make, type, price) VALUES ('Folksvagen', 'Golf', '30000.00');
INSERT 0 1
avocoder=# insert into avto (make, type, price) VALUES ('Folksvagen', 'Passat', '40000.00');
INSERT 0 1

```

```

avocoder=# SELECT * FROM avto;
 id |  make  | type  | price
----+-----+-----+-----
  1 | Reno   | Daster | 25000.00
  2 | Folksvagen | Golf  | 30000.00
  3 | Folksvagen | Passat | 40000.00

```

Рис. 216. Заповнення таблиці *avto*

Додати автомобілі до працівників та обрати перших трьох працівників із компанії (рис. 217).

```

avocoder=# SELECT * FROM employee LIMIT(3);
 id | first_name | last_name | gender | email          | date_of_birth | avto_id
----+-----+-----+-----+-----+-----+-----
  2 | Jone       | Key       | MALE   | kre@gmail.com  | 2014-09-21    |
  3 | Konst      | Lyamar    | MALE   | dpr@gmail.com  | 2015-09-21    |
  1 | Johan      | Krif      | MALE   |                | 2000-01-31    |

```

Рис. 217. Результат обрання перших трьох працівників із автомобілями

Прив'язати **id** автомобіля до кожного співробітника: автомобіль з **id = 1** до співробітника з **id = 2**. Щойно відбуваються зміни, всі вони переносяться в кінець таблиці, тому необхідно обрати за **id**-шником (рис. 218).

```

avecoder=# UPDATE employee SET avto_id = 1 WHERE id = 2;
UPDATE 1
avecoder=# SELECT * FROM employee LIMIT(3);
 id | first_name | last_name | gender |      email      | date_of_birth | avto_id
-----+-----+-----+-----+-----+-----+-----
  3 | Konst      | Lymar     | MALE   | dpr@gmail.com    | 2015-09-21    |
  1 | Johan      | Krif      | MALE   |                 | 2000-01-31    |
  7 | Tatiana    | Montian   | FEMALE | lop@khpi.edu.ua  | 1973-05-03    |
avecoder=# SELECT * FROM employee WHERE id = 2;
 id | first_name | last_name | gender |      email      | date_of_birth | avto_id
-----+-----+-----+-----+-----+-----+-----
  2 | Jone       | Key       | MALE   | kre@gmail.com    | 2014-09-21    |      1

```

Рис. 218. Результат прив'язки **id** автомобіля до співробітників

Контрольні запитання та завдання

1. Чи призведе оновлення таблиці **catalogs** до автоматичного оновлення таблиці **products**?
2. Як зробити унікальним зовнішній ключ?
3. Як проводиться додавання обмеження за допомогою ALTER?
4. Чому взаємопов'язані записи співробітників та автомобілів переносяться в кінець таблиці і як цьому запобігти?

ЛАБОРАТОРНА РОБОТА № 9

РЕАЛІЗАЦІЯ ВНУТРІШНЬОГО ОБ'ЄДНАННЯ ЗА ДОПОМОГОЮ SQL SHELL (КОНСОЛІ)

Мета:

Придбання навичок щодо проведення процедур об'єднання.

Завдання:

1. Провести пряме об'єднання декількох таблиць.
2. Провести об'єднання декількох таблиць за допомогою функції LEFT JOINS.
3. Провести об'єднання декількох таблиць за допомогою функції RIGHT JOINS.
4. Провести об'єднання декількох таблиць за допомогою функції FULL OUTER JOINS.

Виконання завдання 1

Об'єднання декількох таблиць, у яких є однакові колонки (рис. 219).

```
avocoder=# SELECT * FROM employee
avocoder=# JOIN avto ON employee.avto_id = avto.id;
id | first_name | last_name | gender | email | date_of_birth | avto_id | id | make | type | price
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----
2 | Jone | Key | MALE | kre@gmail.com | 2014-09-21 | 1 | 1 | Reno | Daster | 25000.00
(1 ẽẽËÿр)
```

Рис. 219. Результат проведення об'єднання декількох таблиць

Таким чином, за ключем **avto_id** відбулося з'єднання таблиць *employee* і *avto*. Результатом виконання запиту є один запис, оскільки раніше тільки цьому співробітнику було видано автомобіль. Інші співробітники не мають зв'язку з таблицею *avto*, тому в **JOIN** не беруть участі.

Також можна вибирати ті поля, за якими будемо здійснювати об'єднання. Так, наприклад, з таблиці *employee* виберемо поле **first_name**, а з таблиці *avto* – тип. Щоб побачити хто з працівників отримав який автомобіль (рис. 220).

```
avocoder=# SELECT employee.first_name, avto.make, avto.type, avto.price
avocoder=# FROM employee
avocoder=# JOIN avto ON employee.avto_id = avto.id;
first_name | make | type | price
-----+-----+-----+-----
Jone | Reno | Daster | 25000.00
(1 ẽẽËÿр)
```

Рис. 220. Результат перевірки запиту приєднання автомобілів до працівників

Виконання завдання 2

У разі використання цього варіанта до записів першої таблиці додаються записи зі збігами в правій таблиці. Необхідно вказати за яким полем первинний ключ **avto.id** шукатиме збіги в таблиці *employee* за зовнішнім ключем **avto_id** (рис. 221).

```
avocoder=# SELECT * FROM employee
avocoder=# LEFT JOIN avto ON avto.id = employee.avto_id;
```

id	first_name	last_name	gender	email	date_of_birth	avto_id	id	make	type	price
2	Jone	Key	MALE	kre@gmail.com	2014-09-21	1	1	Reno	Daster	25000.00
3	Serhii	Hunter	MALE	hunter@goodle.com	2015-09-21					
9	Johan	Krif	MALE		2000-01-31					
7	Tatiana	Montian	FEMALE	lop@khpi.edu.ua	1973-05-03					

(4 ёёЁюш)

Рис. 221. Результат визначення поля первинного ключа для пошуку збігів в таблиці *employee*

Вивести тільки тих співробітників, у яких є автомобілі (рис. 222).

```
avocoder=# SELECT * FROM employee
avocoder=# LEFT JOIN avto ON avto.id = employee.avto_id WHERE avto_id IS NOT NULL;
```

id	first_name	last_name	gender	email	date_of_birth	avto_id	id	make	type	price
2	Jone	Key	MALE	kre@gmail.com	2014-09-21	1	1	Reno	Daster	25000.00

(1 ёёЁюшр)

Рис. 222. Результат виконання запиту щодо виведення співробітників, у яких є автомобілі за допомогою функції LEFT JOINS

Якщо застосувати **INNER**, то отримаємо такий самий результат (рис. 223).

```
avocoder=# SELECT * FROM employee
avocoder=# INNER JOIN avto ON avto.id = employee.avto_id WHERE avto_id IS NOT NULL;
```

id	first_name	last_name	gender	email	date_of_birth	avto_id	id	make	type	price
2	Jone	Key	MALE	kre@gmail.com	2014-09-21	1	1	Reno	Daster	25000.00

(1 ёёЁюшр)

Рис. 223. Результат виконання запиту щодо виведення співробітників, у яких є автомобілі за допомогою функції LEFT JOINS із застосуванням INNER

Виконання завдання 3

Функція дзеркальна **LEFT JOIN** (рис. 224).

```

avocoder=# SELECT * FROM employee
avocoder=# RIGHT JOIN avto ON avto.id = employee.avto_id;
 id | first_name | last_name | gender | email | date_of_birth | avto_id | id | make | type | price
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----
  2 | Jone | Key | MALE | kre@gmail.com | 2014-09-21 | 1 | 1 | Reno | Daster | 25000.00
    | | | | | | | 2 | Volkswagen | Golf | 30000.00
    | | | | | | | 3 | Volkswagen | Passat | 40000.00
(3 rows)

```

Рис. 224. Результат виконання запиту щодо виведення співробітників, у яких є автомобілі за допомогою функції RIGHT JOINS

Виконання завдання 4

Ця функція об'єднує всі таблиці незалежно від того чи є в них спільні поля (рис. 225).

```

avocoder=# SELECT * FROM employee
avocoder=# FULL OUTER JOIN avto ON avto.id = employee.avto_id;
 id | first_name | last_name | gender | email | date_of_birth | avto_id | id | make | type | price
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----
  2 | Jone | Key | MALE | kre@gmail.com | 2014-09-21 | 1 | 1 | Reno | Daster | 25000.00
    | | | | | | | 2 | Volkswagen | Golf | 30000.00
    | | | | | | | 3 | Volkswagen | Passat | 40000.00
  3 | Serhii | Hunter | MALE | hunter@goodle.com | 2015-09-21 | | | | | | 
  9 | Johan | Krif | MALE | | 2000-01-31 | | | | | | 
  7 | Tatiana | Montian | FEMALE | lop@khpi.edu.ua | 1973-05-03 | | | | | | 
(6 rows)

```

Рис. 225. Результат виконання запиту щодо виведення співробітників, у яких є автомобілі за допомогою функції FULL OUTER JOINS

Контрольні запитання та завдання

1. Як провести пряме об'єднання?
2. Як провести об'єднання за допомогою функції LEFT JOINS?
3. Як провести об'єднання за допомогою функції RIGHT JOINS?
4. Як провести об'єднання за допомогою функції FULL OUTER JOINS?

ЛАБОРАТОРНА РОБОТА № 10

РОЗГОРТАННЯ СУБД FIREBIRD 3.0 З ПІДТРИМКОЮ ШИФРУВАННЯ ДАНИХ ЗАСОБАМИ, РОЗРОБЛЕНИМИ КОМПАНІЄЮ «САЙФЕР»

Мета: встановити та налаштувати СУБД Firebird 3.0 у режимі Super з підтримкою плагінів.

Завдання:

1. Встановити Firebird 3.0.
2. Створити текстову базу даних (БД).
3. Концептуальна схема роботи програмного забезпечення шифрування даних.
4. Генерація контейнера ключа шифрування БД.
5. Інтеграція в СУБД Firebird 3.0 і налаштування плагінів.
6. Шифрування БД.

За допомогою нового об'єктно-орієнтованого C++ API Firebird для розроблення доповнень, що виконуються на боці СУБД, було створено плагіни з реалізацією схем зберігання даних у зашифрованому вигляді. Для ефективного поділу шифрування і доступу до ключа, плагін шифрування бази даних (БД) розділений на дві частини:

- 1) саме шифрування: динамічна бібліотека libCiCryptoStorage.so (CiCryptoStorage.dll);
- 2) тримач секретного ключа: динамічна бібліотека libCiCryptoKeyHolder.so (CiCryptoKeyHolder.dll).

Як полігон обрано операційну систему Debian 8.5.0 x64. Буде створено, зашифровано, модифіковано та розшифровано тестову БД TEST_CRYPT.FDB.

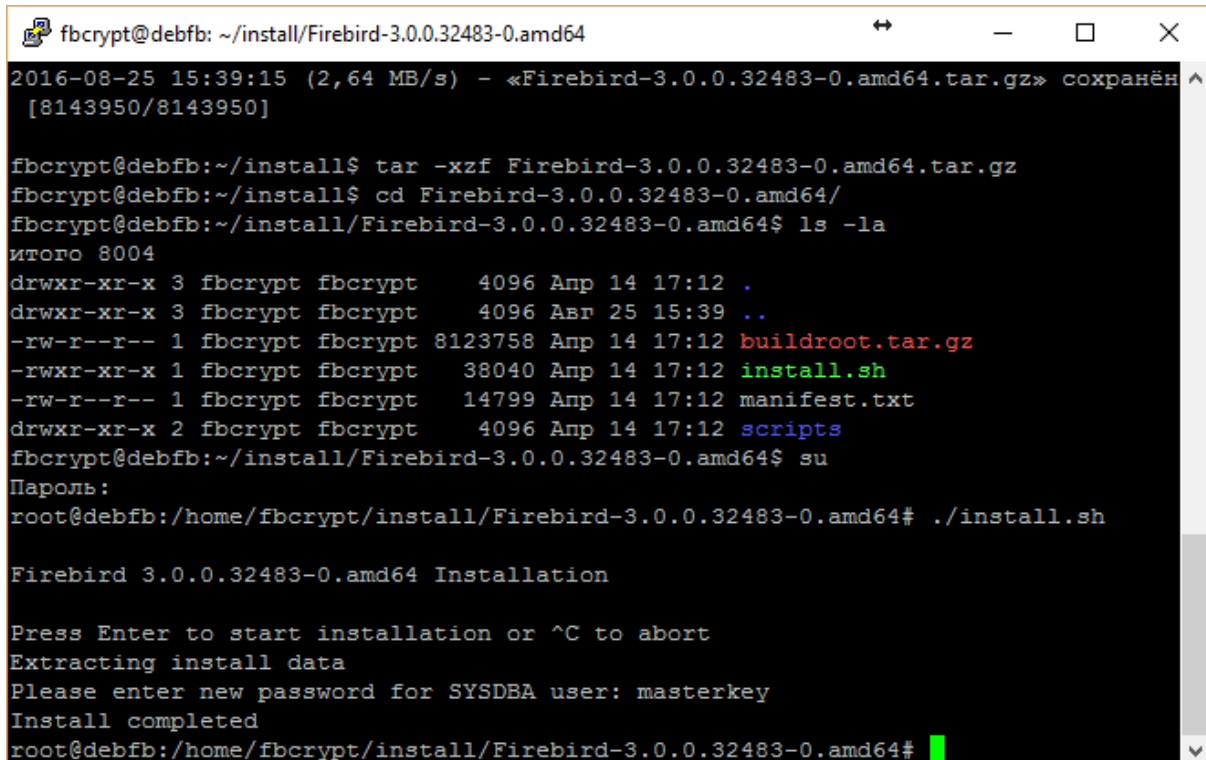
1. Встановлення Firebird 3.0

Скачати з офіційного веб-ресурсу останню актуальну версію Firebird 3.0 і розпакувати:

- fbcrypt@debfb:~/install\$ wget <https://sourceforge.net/projects/firebird/files/firebird-linux-amd64/3.0-Release/Firebird-3.0.0.32483-0.amd64.tar.gz>;
- fbcrypt@debfb:~/install\$ tar -xzf Firebird-3.0.0.32483-0.amd64.tar.gz;
- fbcrypt@debfb:~/install\$ cd Firebird-3.0.0.32483-0.amd64.

Авторизуватися в режимі супер-користувача root та інсталиювати необхідні бібліотеки. Потім запустити встановлення Firebird 3.0 за допомогою скрипта install.sh (рис. 226):

- fbcrypt@debfb:~/install/Firebird-3.0.0.32483-0.amd64\$ su;
- root@debfb:/home/fbcrypt/install/Firebird-3.0.0.32483-0.amd64#apt-get install libtommath-dev;
- root@debfb:/home/fbcrypt/install/Firebird-3.0.0.32483-0.amd64#./install.sh.



```
fbcrypt@debfb: ~/install/Firebird-3.0.0.32483-0.amd64
2016-08-25 15:39:15 (2,64 MB/s) - «Firebird-3.0.0.32483-0.amd64.tar.gz» сохрaнён
[8143950/8143950]

fbcrypt@debfb:~/install$ tar -xzf Firebird-3.0.0.32483-0.amd64.tar.gz
fbcrypt@debfb:~/install$ cd Firebird-3.0.0.32483-0.amd64/
fbcrypt@debfb:~/install/Firebird-3.0.0.32483-0.amd64$ ls -la
итого 8004
drwxr-xr-x 3 fbcrypt fbcrypt 4096 Aug 14 17:12 .
drwxr-xr-x 3 fbcrypt fbcrypt 4096 Aug 25 15:39 ..
-rw-r--r-- 1 fbcrypt fbcrypt 8123758 Aug 14 17:12 buildroot.tar.gz
-rwxr-xr-x 1 fbcrypt fbcrypt 38040 Aug 14 17:12 install.sh
-rw-r--r-- 1 fbcrypt fbcrypt 14799 Aug 14 17:12 manifest.txt
drwxr-xr-x 2 fbcrypt fbcrypt 4096 Aug 14 17:12 scripts
fbcrypt@debfb:~/install/Firebird-3.0.0.32483-0.amd64$ su
Пароль:
root@debfb:/home/fbcrypt/install/Firebird-3.0.0.32483-0.amd64# ./install.sh

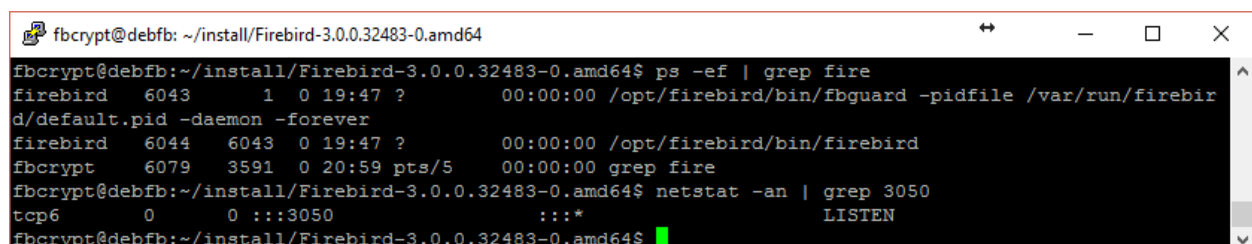
Firebird 3.0.0.32483-0.amd64 Installation

Press Enter to start installation or ^C to abort
Extracting install data
Please enter new password for SYSDBA user: masterkey
Install completed
root@debfb:/home/fbcrypt/install/Firebird-3.0.0.32483-0.amd64#
```

Рис. 226. Процес встановлення Firebird 3.0

Після інсталяції можна перевірити і переконатися, що сервер Firebird запущений і нормально працює (рис. 227):

- fbcrypt@debfb:~/install/Firebird-3.0.0.32483-0.amd64\$ ps -ef | grep fire;
- fbcrypt@debfb:~/install/Firebird-3.0.0.32483-0.amd64\$ netstat -an | grep 3050.



```
fbcrypt@debfb: ~/install/Firebird-3.0.0.32483-0.amd64
fbcrypt@debfb:~/install/Firebird-3.0.0.32483-0.amd64$ ps -ef | grep fire
firebird 6043 1 0 19:47 ? 00:00:00 /opt/firebird/bin/fbguard -pidfile /var/run/firebir
d/default.pid -daemon -forever
firebird 6044 6043 0 19:47 ? 00:00:00 /opt/firebird/bin/firebird
fbcrypt 6079 3591 0 20:59 pts/5 00:00:00 grep fire
fbcrypt@debfb:~/install/Firebird-3.0.0.32483-0.amd64$ netstat -an | grep 3050
tcp6 0 0 :::3050 :::* LISTEN
fbcrypt@debfb:~/install/Firebird-3.0.0.32483-0.amd64$
```

Рис. 227. Перевірка, чи запущено Firebird

Для зручності керування сервером БД Firebird потрібно виконати ще кілька операцій:

- root@debfb:/etc# usermod -a -G firebird fbcrypt;
- root@debfb:/# cp /opt/firebird/misc/firebird.init.d.debian /etc/init.d/firebird;

- root@debfb:/# chmod +x /etc/init.d/firebird;

- root@debfb:/# update-rc.d firebird defaults.

Тепер можна зупиняти, запускати, перезапускати сервер Firebird за допомогою звичайних команд: /etc/init.d/firebird stop|start|restart.

2. Створення тестової БД

Після встановлення та запуску СУБД Firebird можна, використовуючи програмне забезпечення з пакета Firebird (рис. 228):

- fbcrypt@debfb:~\$ /opt/firebird/bin/isql

Use CONNECT or CREATE DATABASE to specify a database

SQL> create database '/home/data/TEST_CRYPT.FDB'

CON> user 'SYSDBA' password 'masterkey';

- SQL> create table t_1 (i1 int, s1 varchar(15), s2 varchar(15));
- SQL> show tables;
- T_1

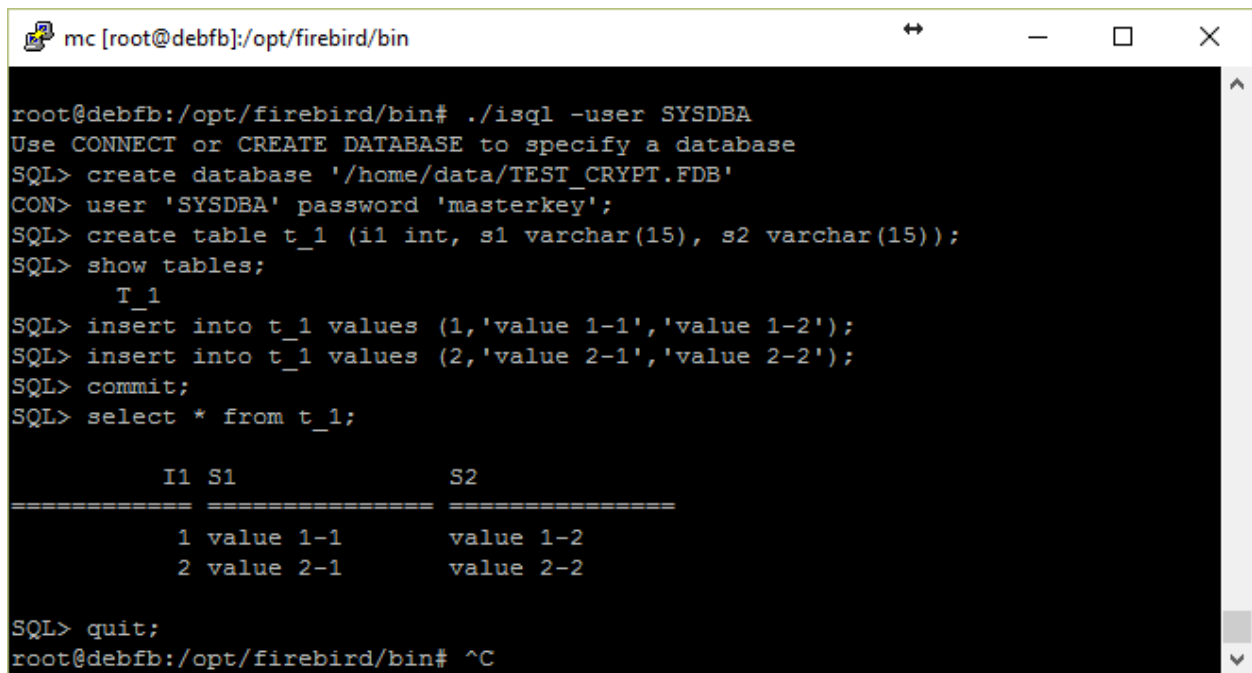
SQL> insert into t_1 values (1,'value 1-1','value 1-2');

- SQL> insert into t_1 values (2,'value 2-1','value 2-2');
- SQL> commit;
- SQL> select * from t_1;
- I1 S1 S2

=====

1	value 1-1	value 1-2
2	value 2-1	value 2-2

SQL> quit;



```
mc [root@debfdb]:/opt/firebird/bin

root@debfdb:/opt/firebird/bin# ./isql -user SYSDBA
Use CONNECT or CREATE DATABASE to specify a database
SQL> create database '/home/data/TEST_CRYPT.FDB'
CON> user 'SYSDBA' password 'masterkey';
SQL> create table t_1 (i1 int, s1 varchar(15), s2 varchar(15));
SQL> show tables;
      T_1
SQL> insert into t_1 values (1,'value 1-1','value 1-2');
SQL> insert into t_1 values (2,'value 2-1','value 2-2');
SQL> commit;
SQL> select * from t_1;

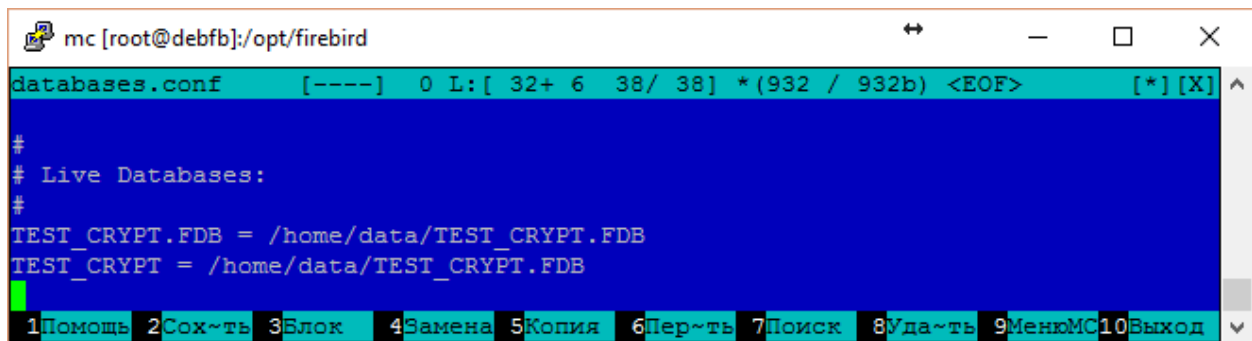
=====
      I1 S1                S2
=====
      1 value 1-1          value 1-2
      2 value 2-1          value 2-2

SQL> quit;
root@debfdb:/opt/firebird/bin# ^C
```

Рис. 228. Процес створення тестової БД

Для доступу до БД віддалено різним програмним забезпеченням зручно використовувати псевдонім, який можна прописати у файлі налаштувань /opt/firebird/databases.conf – наприкінці додати такі рядки (рис. 229):

```
TEST_CRYPT.FDB = /home/data/TEST_CRYPT.FDB
TEST_CRYPT = /home/data/TEST_CRYPT.FDB.
```



```
mc [root@debfdb]:/opt/firebird

databases.conf  [----]  0 L:[ 32+ 6  38/ 38] *(932 / 932b) <EOF>  [*] [X]

#
# Live Databases:
#
TEST_CRYPT.FDB = /home/data/TEST_CRYPT.FDB
TEST_CRYPT = /home/data/TEST_CRYPT.FDB

1Помощь 2Сох~ть 3Блок 4Замена 5Копия 6Пер~ть 7Поиск 8Уда~ть 9МенюMC10Выход
```

Рис. 229. Налаштування псевдоніма тестової БД

Також не варто забувати про права доступу до файлу:
root@debfdb:/home/data# chown firebird:firebird TEST_CRYPT.FDB.

Тепер можна подивитися статус шифрування БД, де можна побачити, що дані не зашифровані (рис. 230): root@debfdb:/opt/firebird/bin#./gstat-e TEST_CRYPT.

```
mc [root@debfb]:/opt/firebird/bin
root@debfb:/opt/firebird/bin# ./gstat -e TEST_CRYPT

Database "/home/data/TEST_CRYPT.FDB"
Database header page information:
  Flags                0
  Generation           11
  System Change Number 0
  Page size            8192
  ODS version          12.0
  Oldest transaction    5
  Oldest active         6
  Oldest snapshot       6
  Next transaction      8
  Sequence number       0
  Next attachment ID    3
  Implementation        HW=AMD/Intel/x64 little-endian OS=Linux CC=gcc
  Shadow count          0
  Page buffers          0
  Next header page      0
  Database dialect      3
  Creation date         Aug 25, 2016 17:53:05
  Attributes            force write

Variable header data:
  *END*

Data pages: total 49, encrypted 0, non-encrypted 49
Index pages: total 57, encrypted 0, non-encrypted 57
Blob pages: total 0, encrypted 0, non-encrypted 0
root@debfb:/opt/firebird/bin#
```

Рис. 230. Просмотр статуса тестовой БД

3. Концептуальна схема роботи програмного забезпечення шифрування даних

Як було зазначено вище, плагін шифрування даних у СУБД Firebird розділений на дві частини. Один із них шифрує дані, а інший тільки зберігає і повертає за запитом першому плагіну ключ шифрування. Природно, якщо перший плагін не отримає ключа з якоїсь причини, операція доступу до даних буде перервана з повідомленням про відсутність ключа.

Плагін, який відповідає за зберігання ключа шифрування БД, своєю чергою, має його десь узяти. У нинішній реалізації ключ зберігається у файлі, але не у відкритому вигляді, а в зашифрованому необхідною кількістю разів. Ця кількість залежить від кількості людей, які беруть участь в адмініструванні і мають право на активацію зашифрованої БД. Оскільки для активації обов'язкова участь двох осіб, ключ шифрування БД також зашифрований ключем шифрування ключа (КЕК), одержуваним за

авторизацією будь-яких двох учасників. Цю кількість можна підрахувати, використовуючи формулу:

$$N = \frac{n(n-1)}{2} \quad (1)$$

де N – отримувана кількість шифрування ключа;

n – кількість учасників.

Для реалізації цієї ідеї було написано окремий застосунок (CiCryptoStarter), що дає можливість сформувати файл-контейнер зашифрованого ключа БД і за необхідності, взаємодіючи з плагіном зберігання ключа, здійснити авторизацію двох осіб, пошук зашифрованого ключа, розшифровування його та передачу плагіну шифрування даних. На рис. 231 показано концептуальну схему взаємодії плагінів і застосунку для забезпечення шифрування даних.

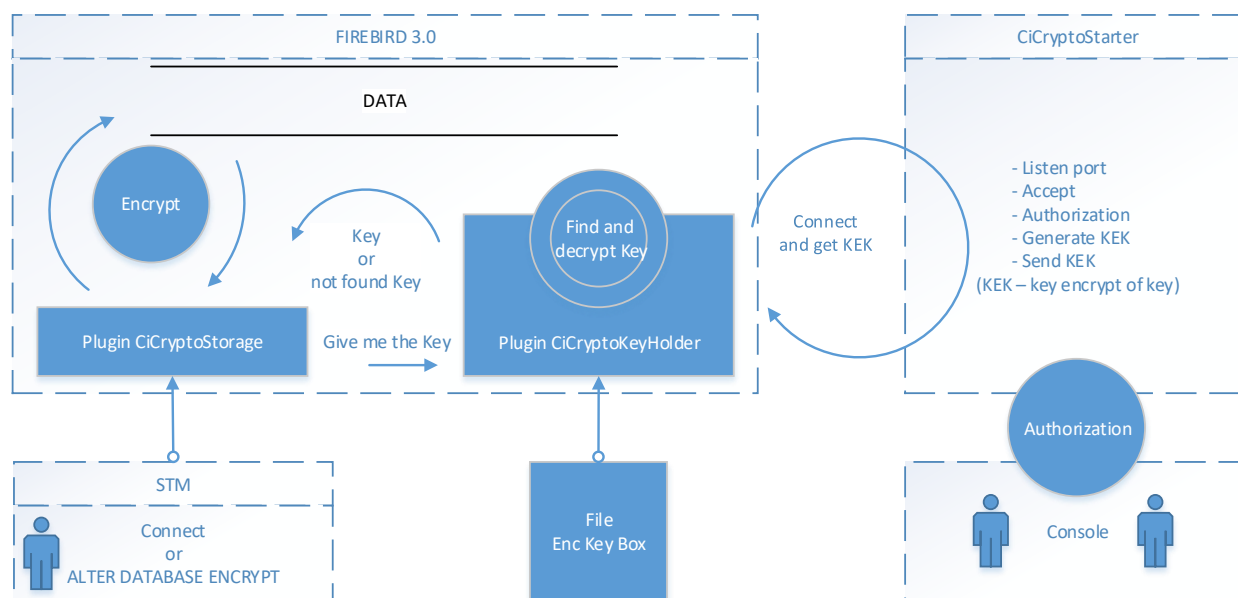


Рис. 231. Концептуальна схема взаємодії плагінів і програми

4. Генерація контейнера ключа шифрування БД

Насамперед необхідно згенерувати файл-контейнер ключа шифрування БД. Це потрібно зробити, використовуючи консольний додаток CiCryptoStarter. На рис. 232 показано довідку з використання цієї програми. Її можна отримати, якщо запустити без параметрів або використовуючи ключ --help або -h.

```
fbcrypt@debf: ~/install
fbcrypt@debf:~/install$ ./CiCryptoStarter
Usage:
$CiCryptoStarter --option [value] [ --option [value] ... ]
Options:
--help(-h)                This description.
--generate(-g) <path to file> Generate key and saving his in file.
--edit(-e) <path to file>   Edit key.
--users(-u) <users count>   The number of users using the key.
--start(-s) <port>         Start application for initialize key DBCrypt.

Example 1:                  CiCryptoStarter --generate /tmp/keybox.dat --users 3
Example 2:                  CiCryptoStarter --edit /tmp/keybox.dat --users 3
Example 3:                  CiCryptoStarter --start 9999

fbcrypt@debf:~/install$
```

Рис. 232. Виведення на консоль довідки програми CiCryptoStarter

Для генерації необхідно вказати ключ `--generate (-g)`. При цьому шлях до файлу можна опустити (файл `keybox.dat` буде створено в цьому самому каталозі) і, якщо в адмініструванні доступу до БД братиме участь лише дві особи (а це мінімальна вимога), то параметр `--users (-u)` також можна не вказувати. Якщо ж необхідна більша кількість осіб, які адмініструють, то слід вказати параметр `--users (-u)` з наступною цифрою кількості учасників. Наприклад, буде використано кількість, що дорівнює трьом (`user1`, `user2`, `user3`) (рис. 233).

```
fbcrypt@debf: ~/install
DB KEY: c1dde0793dba18d9d4c1f00deaeb32e065f88c6484a67eb0c3862fc964e0665c334dbb360efd2
ed90753790042be57fe6588c443ebdad6a064505c3e1a5f4f9f
Enter Username 1: user1
Enter password:

Confirm the password:

Enter Username 2: user2
Enter password:

Confirm the password:

Enter Username 3: user3
Enter password:

Confirm the password:

user1 <---> user2
657b6c8cf79d125f4a3196c3e7fd916b4388de00333c90e5897eb64cf28e73a1
user1 <---> user3
b96efd17edd897c94bc675e5b19ea6439086bb8170abdce9404c16f3a1f727c7
user2 <---> user3
dc15919b1a45859601f7e32656633728d30e658143974c0cc932a0bf53795466
```

Рис. 233. Процес створення файлу-контейнера ключа шифрування БД

Якщо необхідно змінити права доступу до ключа, приміром, коли user2 не повинен мати доступу, а новий user4 повинен, то можна змінити файл-контейнер ключа цим самим застосунком з опцією --edit (-e), після якої вказати шлях до наявного контейнера (якщо не вказати, програма буде його шукати в поточному каталозі під іменем keybox.dat). Також слід вказати параметр --users (-u) з наступною цифрою кількості учасників (у цьому випадку 3). Водночас застосунок спершу запросить авторизацію двох осіб (тут може в авторизації брати участь також і user2, у якого права знімаються), а тільки потім запропонує ввести дані нових учасників, де потрібно буде кожному з них ввести свій логін і пароль із підтвердженням (user1, user3, user4).

5. Інтеграція в СУБД Firebird 3.0 і налаштування плагінів

Коли в наявності вже є сформований файл ключа шифрування БД, можна приступити до інтеграції підтримки шифрування розробленими засобами в СУБД Firebird 3.0. Оскільки за замовчуванням Firebird було проінстальовано в директорію /opt/firebird/, то її і будемо вважати кореневою директорією Firebird. У ній є директорія plugins, куди необхідно помістити наші плагіни libCiCryptoStorage.so і libCiCryptoKeyHolder.so. Тут же, поруч із плагінами, необхідно створити конфігураційні файли плагінів, відповідно CiCryptoStorage.conf (рис. 234) і CiCryptoKeyHolder.conf (рис. 235) з таким вмістом:

```
#      CiCryptoStorage.conf
Plugin_module = CiCryptoStorage {
    Module = $(this)/ CiCryptoStorage
    Config = CiCryptoStorage_conf
}
Plugin_config = CiCryptoStorage_conf {
    AlgCipher = dstu7624-256
    AlgDigest = dstu7564-256
    AlgMode = CFB
}
```

де AlgCipher – алгоритм симетричного шифрування даних;

AlgDigest – алгоритм формування геша;

AlgMode – режим шифрування.

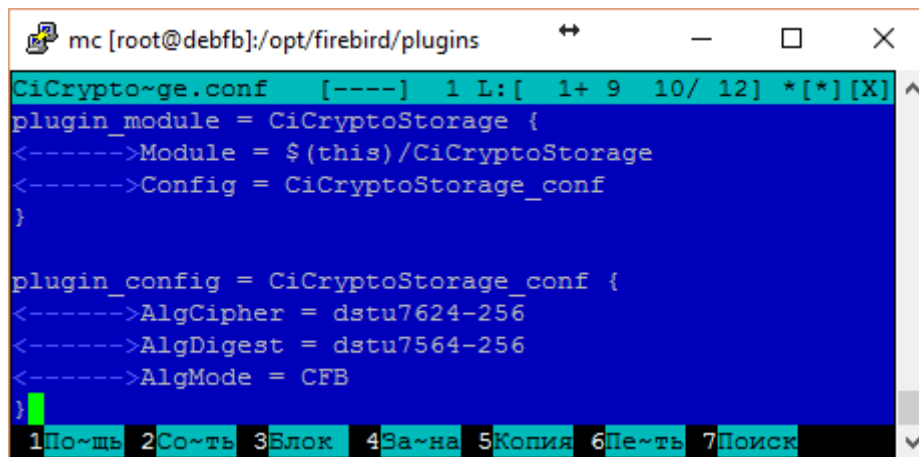


Рис. 234. Конфигурационный файл плагина libCiCryptoStorage

```

#    CiCryptoKeyHolder.conf
Plugin_module = CiCryptoKeyHolder {
    Module = $(this)/CiCryptoKeyHolder
    Config = CiCryptoKeyHolder_conf
}
Plugin_config = CiCryptoKeyHolder_conf {
    Enabled = 1
    Port = 9999
    Path = $(this)/keybox.dat
}

```

де Enabled – прапор увімкнення/вимкнення роботи плагіна;

Port – TCP/IP порт, на якому слухає додаток, що активує CiCryptoStarter;

Path – шлях до файлу-контейнера зашифрованого ключа шифрування БД.

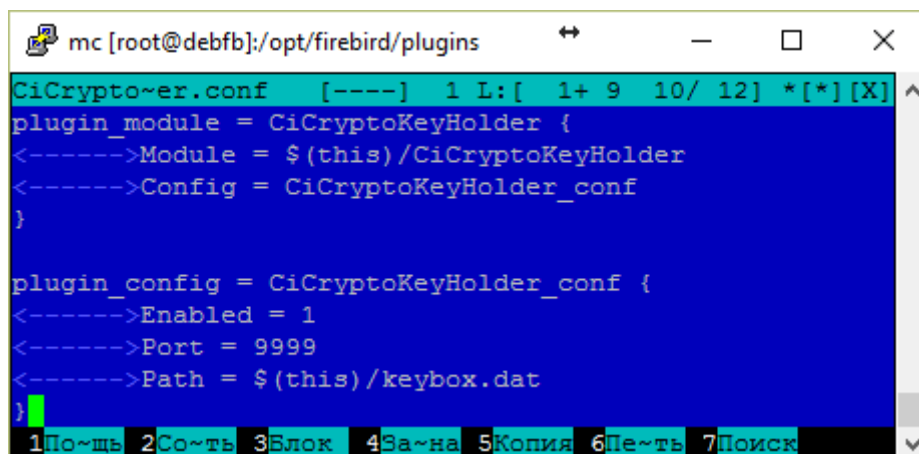


Рис. 235. Конфигурационный файл плагина libCiCryptoKeyHolder

Відповідно до налаштувань плагіна зберігання ключа (рис. 235), у кореневу директорію firebird слід помістити сформований файл ключа шифрування БД keybox.dat.

Для завершення налаштувань залишилося тільки в головному конфігураційному файлі firebird.conf, що міститься в його кореневій директорії firebird, змінити параметр KeyHolderPlugin, вказавши йому значення CiCryptoKeyHolder (рис. 236).

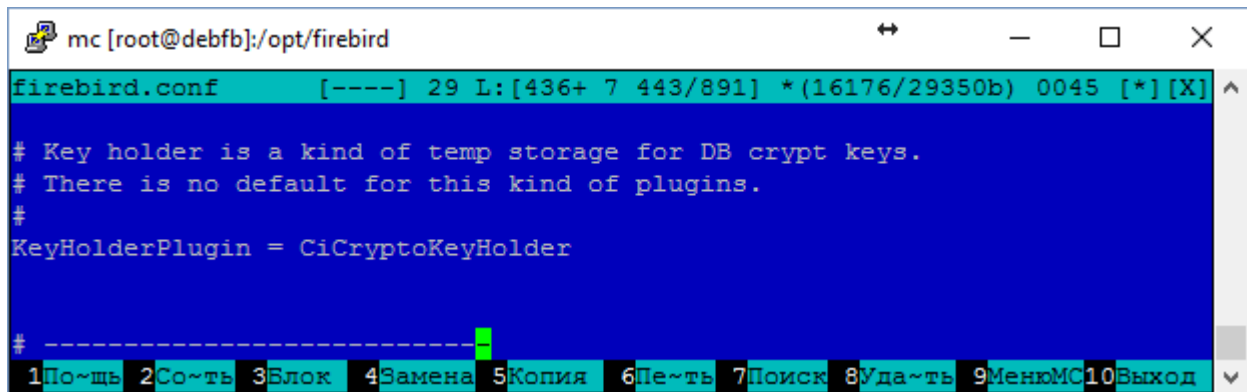


Рис. 236. Конфігураційний файл Firebird

6. Шифрування БД

Тепер можна приступити до шифрування БД. Як видно на рис. 236, під час спроби її зашифрувати та під час підключення до вже зашифрованої, робиться звернення до плагіна шифрування, який, своєю чергою, намагається отримати ключ шифрування у плагіна зберігача ключа. Оскільки останній плагін, якщо ще не отримував ключ, намагається взаємодіяти з додатком CiCryptoStorage, то перш за все необхідно в окремому консольному вікні запустити цей застосунок з опцією --start (-s), після якої слідує числове значення прослуховуваного порту (рис. 237): fbcrypt@debfb:~/install\$./CiCryptoStarter --start 9999.

```
fbcrypt@debf: ~/install
fbcrypt@debf:~/install$ ./CiCryptoStarter --start 9999
----- Summary of socket settings -----
Socket Id:      3
IPv4:           127.0.0.1
port:           9999
debug:          false
reuse addr:     true
keep alive:     false
send buf size:  16384
recv bug size:  87380
blocking:       false
linger on:      false
linger seconds: 0
----- End of Summary of socket settings -----

server: got connection from:
----- Summary of socket settings -----
Socket Id:      4
IPv4:           127.0.0.1
port:           52323
debug:          false
reuse addr:     true
keep alive:     false
send buf size:  2626560
recv bug size:  1062000
blocking:       false
linger on:      false
linger seconds: 0
----- End of Summary of socket settings -----

----- Summary of CiUseCCPP settings -----
AlgID:          7
HashID:         25
ModeID:         13
KeySize:        32
IVSize:         32
is_configured = 1
is_active = 0
----- End of Summary of CiUseCCPP settings -----

Enter Username 1: user2
Enter password:

Enter Username 2: user1
Enter password:

user2 <---> user1
KEK: 657b6c8cf79d125f4a3196c3e7fd916b4388de00333c90e5897eb64cf28e73a1
SALT: 7660a2cb539a0c47df6c9b6444cefc688dd256a169a17e373751bc733dbc928a7816ceadd57f
GET PBKDF: salt = v`S
G1dV~77Qs=xE pass = 2016/08/29 00:00:00
DATE PBKDF: 2b5c3cfa4aa9df70ff90132c27c9641adbd6399315949032a48ed56228f75980
EKEK: e1a18c0e01f3184f4669d1c8d84d18418a1ef81706e129c6b120ecc165471371cc1d41d005e7f35b3b4
0766c801ad316199162bd227
Send KEK OK !
OK !!! Len = 64
```

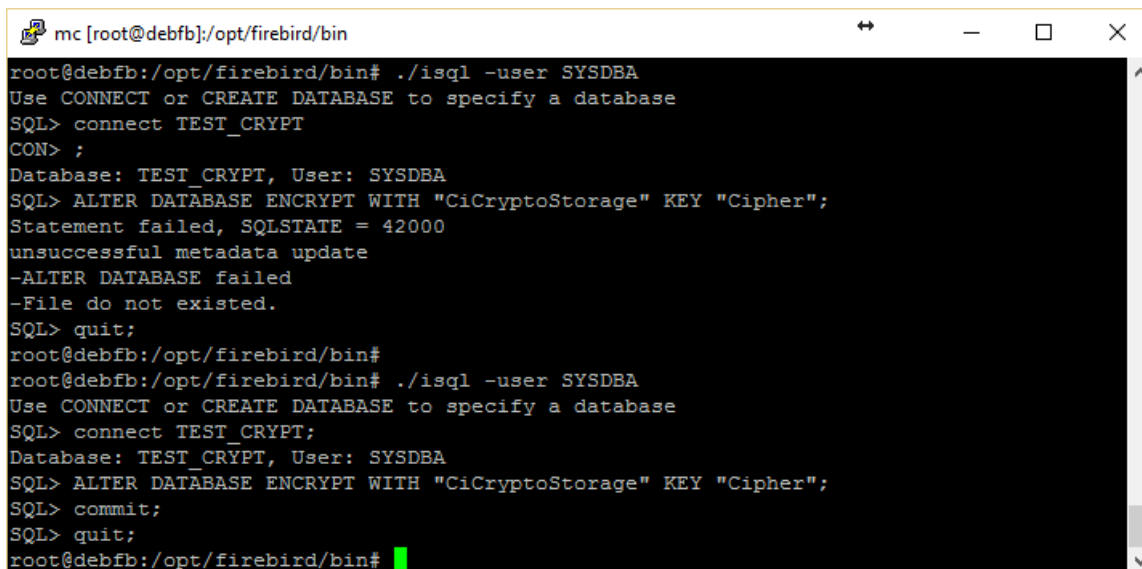
Рис. 237. Додаток CiCryptoStarter у режимі активації ключа

Зашифрувати БД можна як локально, використовуючи програмне забезпечення пакета firebird (рис. 238), так і віддалено (рис. 239), підключившись до БД і виконавши запит на зміну її налаштувань (рис. 240):

- root@debf:/opt/firebird/bin# ./isql -user SYSDBA
Use CONNECT or CREATE DATABASE to specify a database
SQL> connect TEST_CRYPT;
- Database: TEST_CRYPT, User: SYSDBA

SQL> ALTER DATABASE ENCRYPT WITH "CiCryptoStorage" KEY "Cipher";

- SQL> commit;
- SQL> quit;



```
mc [root@debfdb]:/opt/firebird/bin
root@debfdb:/opt/firebird/bin# ./isql -user SYSDBA
Use CONNECT or CREATE DATABASE to specify a database
SQL> connect TEST_CRYPT
CON> ;
Database: TEST_CRYPT, User: SYSDBA
SQL> ALTER DATABASE ENCRYPT WITH "CiCryptoStorage" KEY "Cipher";
Statement failed, SQLSTATE = 42000
unsuccessful metadata update
-ALTER DATABASE failed
-File do not existed.
SQL> quit;
root@debfdb:/opt/firebird/bin#
root@debfdb:/opt/firebird/bin# ./isql -user SYSDBA
Use CONNECT or CREATE DATABASE to specify a database
SQL> connect TEST_CRYPT;
Database: TEST_CRYPT, User: SYSDBA
SQL> ALTER DATABASE ENCRYPT WITH "CiCryptoStorage" KEY "Cipher";
SQL> commit;
SQL> quit;
root@debfdb:/opt/firebird/bin#
```

Рис. 238. Зашифровування БД за допомогою isql

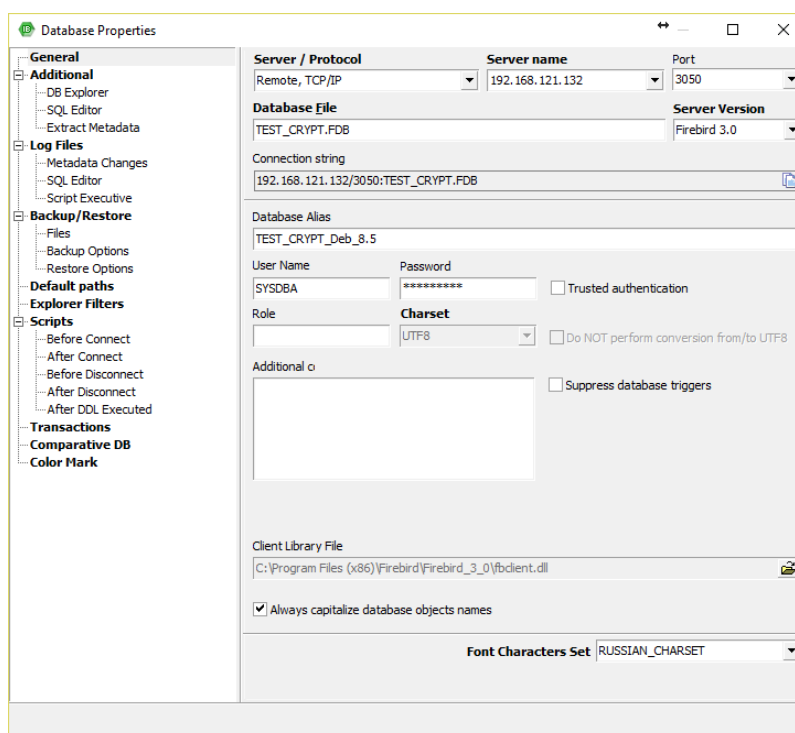


Рис. 239. Налаштування віддаленого підключення в програмі ІВЕксперт

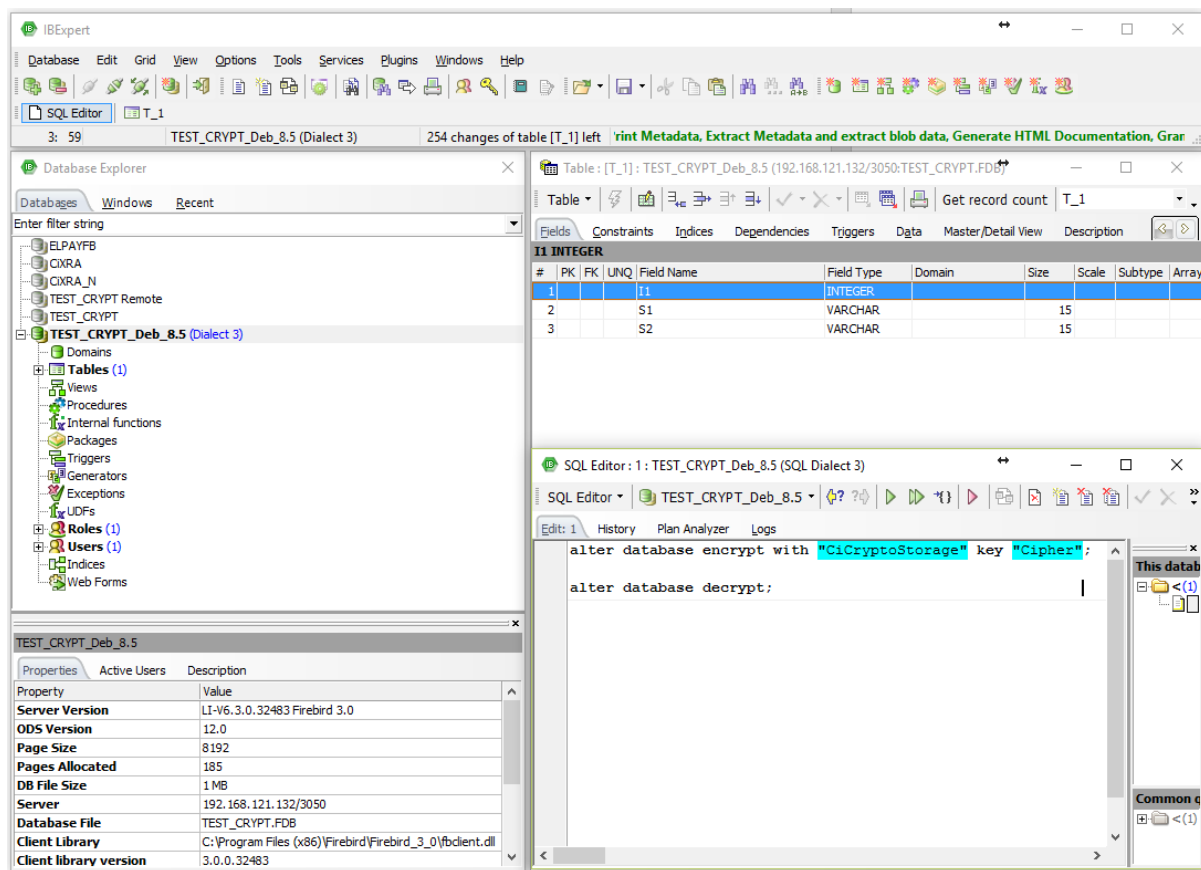


Рис. 240. Управління налаштуваннями БД у програмі ІВЕксперт

Після виконання запиту і фіксації його в БД можна знову звернутися до програми виведення статусу шифрування БД (рис. 241).

```
mc [root@debfdb]:/opt/firebird/bin

root@debfdb:/opt/firebird/bin# ./gstat -e TEST_CRYPT

Database "/home/data/TEST_CRYPT.FDB"
Database header page information:
  Flags                0
  Generation           20
  System Change Number 0
  Page size            8192
  ODS version          12.0
  Oldest transaction    11
  Oldest active         12
  Oldest snapshot      12
  Next transaction      15
  Sequence number       0
  Next attachment ID    10
  Implementation        HW=AMD/Intel/x64 little-endian OS=Linux CC=gcc
  Shadow count          0
  Page buffers          0
  Next header page      0
  Database dialect      3
  Creation date         Aug 25, 2016 17:53:05
  Attributes            force write, encrypted, plugin CiCryptoStorage

Variable header data:
  Key hash:             PxYyKpVSOWcXNWco11b3PCFdaS8=
  Encryption key name:   Cipher
  *END*

Data pages: total 49, encrypted 49, non-crypted 0
Index pages: total 57, encrypted 57, non-crypted 0
Blob pages: total 0, encrypted 0, non-crypted 0
root@debfdb:/opt/firebird/bin#
```

Рис. 241. Перегляд статусу тестової БД

Тепер видно, що дані зашифровані ключем на ім'я “Cipher”.

До перезавантаження сервера БД Firebird, який працює в режимі Super, ключ активувати більше не потрібно. Можна налаштувати і підключитися до зашифрованої БД будь-якою з програм і працювати з даними. Наприклад, було встановлено на Debian програму FlameRobin і організовано віддалене підключення до зашифрованої тестової БД (рис. 242).

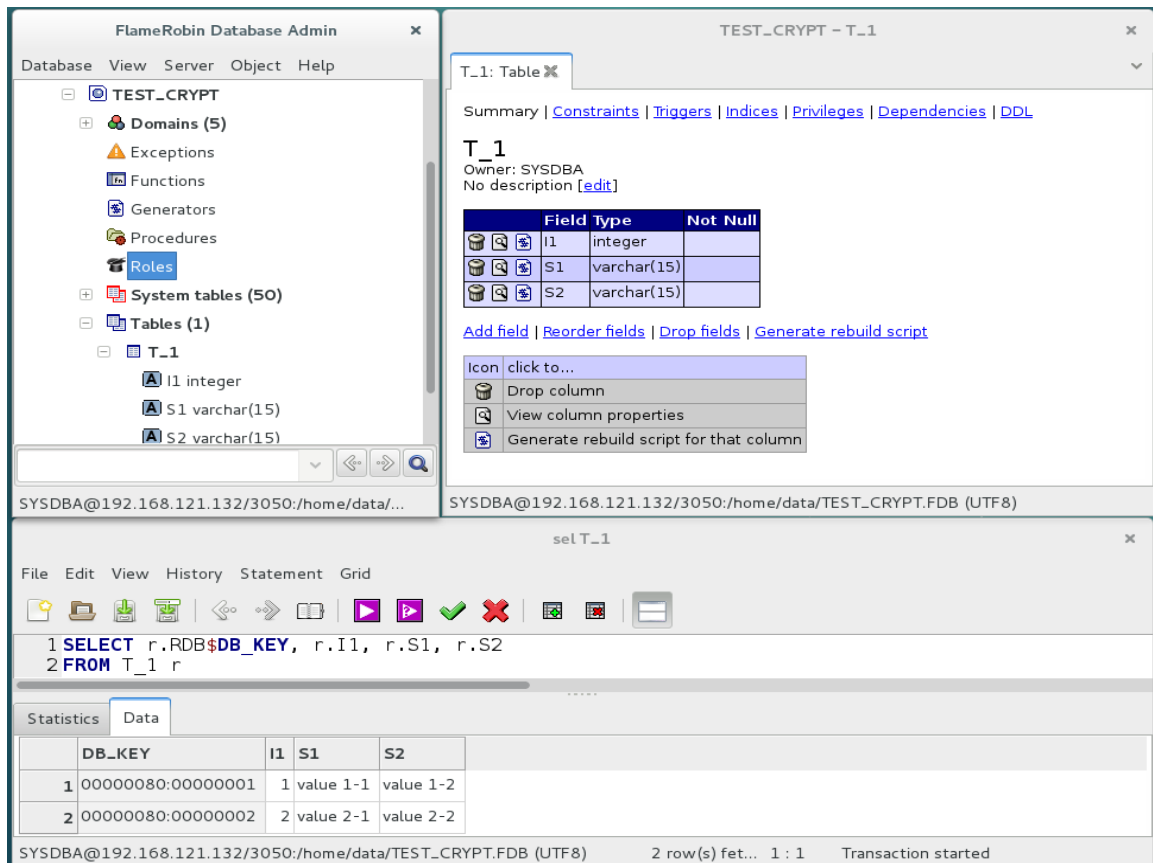


Рис. 242. Віддалене підключення до тестової БД у програмі FlameRobin

У будь-якому з підключень (звісно, все залежить від прав до БД) також можна розшифрувати тестову БД (рис. 243).

```
mc [root@debfb]:/opt/firebird/bin
root@debfb:/opt/firebird/bin# ./gstat -e TEST_CRYPT

Database "/home/data/TEST_CRYPT.FDB"
Database header page information:
  Flags                0
  Generation           69
  System Change Number 0
  Page size            8192
  ODS version          12.0
  Oldest transaction    50
  Oldest active         51
  Oldest snapshot       51
  Next transaction      62
  Sequence number       0
  Next attachment ID    19
  Implementation        HW=AMD/Intel/x64 little-endian OS=Linux CC=gcc
  Shadow count          0
  Page buffers          0
  Next header page      0
  Database dialect      3
  Creation date         Aug 25, 2016 17:53:05
  Attributes            force write

Variable header data:
  *END*

Data pages: total 49, encrypted 0, non-crypted 49
Index pages: total 57, encrypted 0, non-crypted 57
Blob pages: total 0, encrypted 0, non-crypted 0
root@debfb:/opt/firebird/bin#
```

Рис. 243. Перегляд статусу тестової БД

Для цього достатньо виконати запит і його фіксацію: ALTER DATABASE DECRYPT.

Контрольні запитання та завдання

1. Що забезпечує використання СУБД Firebird 3.0?
2. Як встановити статус шифрування БД?
3. За допомогою чого налаштовується безпека віддалене підключення до БД?

ЗАВДАННЯ ДЛЯ ВИКОНАННЯ ПІДСУМКОВОЇ РОБОТИ

Варіант 1

1. Вивести імена всіх людей, які є в базі даних авіакомпаній (Поля в результуючій таблиці: name).
2. Вивести назви всіх авіакомпаній (Поля в результуючій таблиці: name).
3. Вивести всі рейси, здійснені з Лондона (Поля в результуючій таблиці: *).
4. У які міста можна полетіти з Парижа (Paris) і скільки часу це займе? (Поля в результуючій таблиці: town_to, flight_time. Використовуйте конструкцію "as flight_time" для виведення необхідного часу. Це необхідно для коректної перевірки. Формат для виведення часу: HH:MM:SS).
5. Вивести вильоти, здійснені з 10 год. по 14 год. 1 січня 2024 р. (Поля в результуючій таблиці: *).
6. Видалити компанії, які здійснили найменшу кількість рейсів.

Варіант 2

1. Скільки всього 10-их класів? (Поля в результуючій таблиці: count. Використовуйте конструкцію "as count" для агрегатної функції підрахунку кількості класів. Це необхідно для коректної перевірки).
2. Виведіть інформацію про учнів, які живуть на вулиці Сумській (ul. Sumska)? (Поля в результуючій таблиці: *)
3. Скільки Ганн (Ganna) навчається в школі? (Поля в результуючій таблиці: count. Використовуйте конструкцію "as count" для агрегатної функції підрахунку кількості учнів. Це необхідно для коректної перевірки).
4. Скільки різних кабінетів школи використовувалися 2 вересня 2024 року для проведення занять? (Поля в результуючій таблиці: count. Використовуйте конструкцію "as count" для агрегатної функції підрахунку кількості різних кабінетів. Це необхідно для коректної перевірки).
5. Скільки років наймолодшому учню, який навчається? (Поля в результуючій таблиці: year. Використовуйте конструкцію "as year" для зазначення віку учня. Це необхідно для коректної перевірки).

6. Знайдіть максимальний вік (кількість років) серед учнів 10 класів на сьогоднішній день. Для отримання поточних дати та часу використовуйте функцію NOW(). (Поля в результуючій таблиці: max_year. Використовуйте конструкцію "as max_year" для зазначення максимального віку в роках. Це необхідно для коректної перевірки).

Варіант 3

1. Визначити, хто з членів сім'ї купував картоплю (potato) (Поля в результуючій таблиці: status).

2. Знайти імена всіх матерів (mother) (Поля в результуючій таблиці: member_name).

3. Змініть ім'я "Andie Quincey" на нове "Andie Anthony".

4. Визначити, скільки витратив у 2023 році кожен із членів сім'ї. У результуючій вибірці не виводьте тих членів сім'ї, які нічого не витратили. (Поля в результуючій таблиці: member_name, status, costs. Використовуйте конструкцію "as costs" для відображення витраченої суми членом сім'ї. Це необхідно для коректної перевірки).

5. Виведіть ім'я найстаршої людини. Якщо таких кілька, то виведіть їх усіх. (Поля в результуючій таблиці: member_name).

6. Скільки і хто з родини витратив на розваги (entertainment). Вивести статус у сім'ї, ім'я, суму. (Поля в результуючій таблиці: status, member_name, costs. Використовуйте конструкцію "as costs" для відображення витраченої суми членом сім'ї. Це необхідно для коректної перевірки).

Варіант 4

1. Виведіть ідентифікатор і ознаку наявності інтернету в приміщенні. Якщо інтернет у помешканні, що здається, присутній, то виведіть «YES», інакше «NO». (Поля в результуючій таблиці: id, has_internet. Використовуйте конструкцію "AS has_internet" для виведення ознаки наявності інтернету в приміщенні).

Формат очікуваного результату запиту:

	id	has_internet
1	1	NO
2	2	YES

2. Вивести користувачів, які вказали Білоруський номер телефону. Телефонний код Білорусі +375. (Поля в результуючій таблиці: *)

3. Вибрати всі записи, в яких рік верифікації email 2021 р.

4. Виведіть список кімнат, які були зарезервовані хоча б на одну добу в 12-й тиждень 2024 року. У цій задачі за один тиждень візьміть період із семи днів, перший з яких починається 1 січня 2024 року. Наприклад, перший тиждень року – 1-7 січня, а третій – 15-21 січня. (Поля в результуючій таблиці: Rooms.*)

5. Вивести в порядку убутання популярності доменні імена 2-го рівня, що використовуються користувачами для електронної пошти. Отриманий результат необхідно додатково відсортувати за зростанням назв доменних імен. (Поля в результуючій таблиці: domain, count. Для ел. пошти index@gmail.com доменним ім'ям 2-го рівня буде gmail.com).

6. Додати відгук із рейтингом 5 на житло, що знаходяться за адресою "11218, Friel Place, New York", від імені "George Clooney" (В якості первинного ключа (id) вкажіть кількість записів у таблиці + 1. Резервування кімнати, на яку вам потрібно залишити відгук, уже було зроблено, потрібно лише її знайти).

Варіант 5

1. Вивести імена людей, які закінчуються на "man" (Поля в результуючій таблиці: name).

2. Вивести кількість рейсів, здійснених на Ербас А320-200 (Поля в результуючій таблиці: count. Використовуйте конструкцію "as count" для агрегатної функції підрахунку кількості рейсів. Це необхідно для коректної перевірки).

3. Які компанії здійснювали перельоти на Boeing (Поля в результуючій таблиці: name)

4. Виведіть пасажирів із найдовшим ПІБ. Пробіли, дефіси та крапки вважаються частиною імені. (Поля в результуючій таблиці: name).

5. Вивести імена людей, у яких є повний тезка серед пасажирів (Поля в результуючій таблиці: name).

6. Видалити компанії, які здійснили найменшу кількість рейсів.

Варіант 6

1. Скільки учнів у 10 В класі? (Поля в результуючій таблиці: count. Використовуйте конструкцію "as count" для агрегатної функції підсчета количества учащихся. Это необходимо для корректной проверки).

2. З'ясуйте, о котрій годині за розкладом починається четверте заняття.

3. (Поля в результуючій таблиці: start_pair).

4. Виведіть прізвище, ім'я та дату народження учнів, хто був народжений у травні. (Поля в результуючій таблиці: last_name, first_name, birthday).

5. Виведіть назву предметів, які викладає Ромашкін П.П. (Romashkin P.P.). Зверніть увагу, що в базі даних є кілька вчителів із такими прізвищем та ініціалами. (Поля в результуючій таблиці: subjects. Використовуйте конструкцію "as subjects" для зазначення навчальних предметів. Це необхідно для коректної перевірки).

6. Скільки часу учень перебуватиме в школі, навчаючись з 2-го по 4-ий уч. предмет? (Поля в результуючій таблиці: time. Використовуйте конструкцію "as time" для зазначення різниці в часі. Це необхідно для коректної перевірки. Результат має бути у форматі HH:MM:SS).

7. Які кабінети найчастіше використовувалися для проведення занять? Виведіть ті, які використовувалися максимальну кількість разів. (Поля в результуючій таблиці: classroom).

Варіант 7

1. Визначити, хто з членів сім'ї купував картоплю (potato) (Поля в результуючій таблиці: status).

2. Знайти імена всіх матерів (mother) (Поля в результуючій таблиці: member_name).

3. Змініть ім'я "Andie Quincey" на нове "Andie Anthony".

4. Визначити товари, які купували більше 1 разу. (Поля в результуючій таблиці: good_name).

5. Знайдіть найдорожчий делікатес (delicacies) і виведіть його ціну (Поля в результуючій таблиці: good_name, unit_price).

6. Визначити хто і скільки витратив у червні 2023 (Поля в результуючій таблиці: member_name costs. Використовуйте конструкцію "as costs" для відображення витраченої суми членом сім'ї. Це необхідно для коректної перевірки).

Варіант 8

1. Виведіть ідентифікатор і ознаку наявності кондиціонеру в приміщенні. Якщо кондиціонер у помешканні, що здається, присутній, то

виведіть "YES", інакше "NO". (Поля в результуючій таблиці: id, has_air_con. Використовуйте конструкцію "AS has_air_con" для виведення ознаки наявності кондиціонера в приміщенні.

Формат очікуваного результату запиту:

	id	has_air_con
1	1	NO
2	2	YES

2. Вивести користувачів, які вказали +994. (Поля в результуючій таблиці: *).

3. Вибрати всі записи, в яких рік верифікації email 2022 р.

4. Вивести кількість бронювань за кожним місяцем кожного року, у яких було хоча б 1 бронювання. Результат відсортуйте в порядку зростання дати бронювання (Поля в результуючій таблиці: year, month, amount. Використовуйте конструкції "as year", "as month" та "as amount" для виведення року і місяця бронювання, кількості таких бронювань відповідно).

Формат очікуваного результату запиту:

	year	month	amount
1	2021	7	8
2	2022	10	2
3	2023	11	1

5. Необхідно вивести рейтинг для кімнат, які хоч раз орендували, як середнє значення рейтингу відгуків, округлене до цілого за зниженням. (Поля в результуючій таблиці: room_id, rating. Іспользуйте конструкцію "as rating" для вивода рейтинга).

6. Для кожної кімнати, яку винаймали щонайменше 1 раз, знайдіть ім'я людини, яка винаймала її востаннє, і дату, коли вона виїхала (Поля в результуючій таблиці: room_id, name, end_date. Використовуйте конструкцію "as room_id" для виведення ідентифікатора кімнати).

Варіант 9

1. Вивести всі назви літаків, на яких можна полетіти до Львову (Lviv) (Поля в результуючій таблиці: plane).

2. Які компанії організовують перельоти з Києва (Kyiv)? (Поля в результуючій таблиці: name).

3. Виведіть ідентифікатори всіх рейсів і кількість пасажирів на них. Зверніть увагу, що на якихось рейсах пасажирів може не бути. У цьому

випадку виведіть число "0". (Поля в результуючій таблиці: id, count. Використовуйте конструкцію "as count" для агрегатної функції підрахунку кількості рейсів. Це необхідно для коректної перевірки).

4. Вивести відсортований за кількістю перельотів (за зменшенням) та ім'ям (за зростанням) список пасажирів, які здійснили хоча б 1 політ. (Поля в результуючій таблиці: name, count. Використовуйте конструкцію "as count" для агрегатної функції підрахунку кількості перельотів. Це необхідно для коректної перевірки).

5. Виведіть імена пасажирів, які полетіли до Києва (Kyiv) на літаку Ембраер (Поля в результуючій таблиці: name).

6. Видалити компанії, які здійснили найменшу кількість рейсів.

Варіант 10

1. Скільки всього 10-их класів? (Поля в результуючій таблиці: count. Використовуйте конструкцію "as count" для агрегатної функції підрахунку кількості класів. Це необхідно для коректної перевірки).

2. Скільки Ганн (Ganna) навчається в школі? (Поля в результуючій таблиці: count. Використовуйте конструкцію "as count" для агрегатної функції підрахунку кількості учнів. Це необхідно для коректної перевірки).

3. Скільки учнів у 10 В класі? (Поля в результуючій таблиці: count. Використовуйте конструкцію "as count" для агрегатної функції підрахунку кількості учнів. Це необхідно для коректної перевірки).

4. Виведіть прізвища викладачів, які ведуть фізичну культуру (Physical Culture). Відсортуйте викладачів за прізвищами в алфавітному порядку. (Поля в результуючій таблиці: last_name).

5. У яких класах запровадить заняття викладач "Krauze"? (Поля в результуючій таблиці: name).

6. Виведіть ідентифікатори викладачів, які хоча б один раз за весь час викладали в кожному з одинадцятих класів. (Поля в результуючій таблиці: teacher).

Варіант 11

1. Визначити, хто з членів сім'ї купував картоплю (potato) (Поля в результуючій таблиці: status).

2. Знайти імена всіх матерів (mother) (Поля в результуючій таблиці: member_name).

3. Змініть ім'я "Andie Quincey" на нове "Andie Anthony".

4. Визначити, які товари не купували у 2023 році (Поля в результуючій таблиці: good_name. Усі доступні до купівлі продукти знаходяться в таблиці Goods).

5. Визначити групи товарів, які не купувалися у 2023 році (Поля в результуючій таблиці: good_type_name).

6. Дізнайтеся, скільки було витрачено на кожну з груп товарів у 2023 році. Виведіть назву групи та витрачену на неї суму. Якщо витрачена сума дорівнює нулю, тобто товари з цієї групи не купувалися у 2023 році, то не виводьте її (Поля в результуючій таблиці: good_type_name costs. Використовуйте конструкцію "as costs" для відображення витраченої суми на конкретну групу товарів. Це необхідно для коректної перевірки).

Варіант 12

1. Виведіть ідентифікатор і ознаку наявності інтернету в приміщенні. Якщо інтернет у помешканні, що здається, присутній, то виведіть «YES», інакше «NO». (Поля в результуючій таблиці: id, has_internet. Використовуйте конструкцію "AS has_internet" для виведення ознаки наявності інтернету в приміщенні).

Формат очікуваного результату запиту:

	id	has_internet
1	1	NO
2	2	YES

2. Вивести користувачів, які вказали +43. (Поля в результуючій таблиці: *).

3. Вибрати всі записи, в яких пропонуються до оренди кімнати (Private room).

4. Вивести список кімнат з усіма зручностями (наявність ТБ, інтернету, кухні та кондиціонера), а також загальну кількість днів і суму за всі дні оренди кожної з таких кімнат. (Поля в результуючій таблиці: home_type, address, days, total_fee. Використовуйте конструкції "as days" та "as total_fee" для виведення кількості днів і суми оренди, відповідно. Якщо кімната не здавалася, то кількість днів і суму вивести як 0).

5. Необхідно категоризувати житло на есоному, comfort, premium за ціною відповідно ≤ 100 , $100 < \text{ціна} < 200$, ≥ 200 . Як результат вивести таблицю з назвою категорії та кількістю житла, що потрапляє в цю категорію (Поля в результуючій таблиці: category, count. Використовуйте

конструкції "as category" та "as count" для виведення назви категорії та кількості такого житла відповідно).

Формат очікуваного результату запиту:

	category	count
1	economy	1
2	comfort	2
3	premium	3

6. Вивести ідентифікатори всіх власників кімнат, що розміщені на сервісі бронювання житла та суму, яку вони заробили (Поля в результуючій таблиці: owner_id, total_earn. Використовуйте конструкції "as owner_id" та "as total_earn" для виведення ідентифікаторів власників і заробленої суми відповідно).

Варіант 13

1. Вивести імена всіх людей, які є в базі даних авіакомпаній (Поля в результуючій таблиці: name).

2. У які міста літав Bruce Willis (Поля в результуючій таблиці: town_to).

3. Виведіть дату та час прильоту пасажира Стів Мартін (Steve Martin) в Лондон (London) (Поля в результуючій таблиці: time_in).

4. Виведіть навантаженість (кількість пасажирів) кожного рейсу (trip). Результат вивести у відсортованому вигляді за спаданням навантаженості. (Поля в результуючій таблиці: trip, count. Використовуйте конструкцію "as count" для агрегатної функції підрахунку кількості пасажирів на рейсі. Це необхідно для коректної перевірки).

5. Вивести час відльоту і час прильоту для кожного перельоту у форматі "ГГ:XX, ДД.ММ – ГГ:XX, ДД.ММ", де години і хвилини з провідним нулем, а день і місяць без. (Поля в результуючій таблиці: flight_time. Використовуйте конструкцію "as flight_time" для отриманого рядка з датами відльоту і прильоту).

Формат очікуваного результату запиту:

	flight_time
1	11:25, 3.11 - 16:50, 3.11
2	10:00, 3.1 - 11:00, 3.1

6. Видалити компанії, які здійснили найменшу кількість рейсів.

Варіант 14

1. Скільки всього 10-их класів? (Поля в результуючій таблиці: count. Використовуйте конструкцію "as count" для агрегатної функції підрахунку кількості класів. Це необхідно для коректної перевірки).

2. Скільки учнів у 10 В класі? (Поля в результуючій таблиці: count. Використовуйте конструкцію "as count" для агрегатної функції підрахунку кількості учнів. Це необхідно для коректної перевірки).

3. З'ясуйте, о котрій годині за розкладом починається четверте заняття. (Поля в результуючій таблиці: start_pair).

4. Скільки занять провів Krauze 30 серпня 2024 р.? (Поля в результуючій таблиці: count. Використовуйте конструкцію "as count" для агрегатної функції підрахунку числа занять. Це необхідно для коректної перевірки).

5. Виведіть заповненість класів у порядку убавання (Поля в результуючій таблиці: name, count. Використовуйте конструкцію "as count" для агрегатної функції підрахунку кількості учнів у класах. Це необхідно для коректної перевірки).

6. Знайдіть максимальний вік (кількість років) серед учнів 10 класів на сьогоднішній день. Для отримання поточних дати та часу використовуйте функцію NOW(). (Поля в результуючій таблиці: max_year. Використовуйте конструкцію "as max_year" для зазначення максимального віку в роках. Це необхідно для коректної перевірки).

Варіант 15

1. Визначити, хто з членів сім'ї купував картоплю (potato) (Поля в результуючій таблиці: status).

2. Знайти імена всіх матерів (mother) (Поля в результуючій таблиці: member_name).

3. Змініть ім'я "Andie Quincey" на нове "Andie Anthony".

4. Вивести всіх членів сім'ї з прізвищем Quincey. (Поля в результуючій таблиці: *).

5. Вивести середній вік людей (у роках), що зберігаються в базі даних. Результат округліть до цілого в менший бік. (Поля в результуючій таблиці: age. Використовуйте конструкцію "as age" для агрегатної функції підрахунку середнього віку. Це необхідно для коректної перевірки).

6. Знайдіть середню ціну ікри на основі даних, що зберігаються в таблиці Payments. У базі даних зберігаються дані про покупки червоної

(red caviar) і чорної ікри (black caviar). У відповіді має бути один рядок із середньою ціною всієї купленої коли-небудь ікри. (Поля в результуючій таблиці: cost. Використовуйте конструкцію "as cost" для агрегатної функції підрахунку середньої ціни ікри. Це необхідно для коректної перевірки).

Варіант 16

1. Виведіть ідентифікатор і ознаку наявності кондиціонеру в приміщенні. Якщо кондиціонер у помешканні, що здається, присутній, то виведіть "YES", інакше "NO" (Поля в результуючій таблиці: id, has_air_con. Використовуйте конструкцію "AS has_air_con" для виведення ознаки наявності кондиціонеру в приміщенні).

Формат очікуваного результату запиту:

	id	has_air_con
1	1	NO
2	2	YES

2. Вивести користувачів, які вказали +61. (Поля в результуючій таблиці: *).

3. Вибрати всі записи, в яких рік верифікації email 2024 р.

4. Виведіть середню ціну бронювання за добу для кожної з кімнат, яку бронювали хоча б один раз. Середню ціну необхідно округлити до цілого значення вгору (Поля в результуючій таблиці: room_id, avg_price. Використовуйте конструкцію "as avg_price" для виведення середньої вартості бронювання для кімнат).

5. Виведіть id тих кімнат, які орендували непарну кількість разів (Поля в результуючій таблиці: room_id, count. Використовуйте конструкцію "as count" для виведення кількості скільки разів кімнату брали в оренду).

6. Знайдіть який відсоток користувачів, зареєстрованих на сервісі бронювання, хоч раз орендували або здавали в оренду житло. Результат округліть до сотих (Поля в результуючій таблиці: percent. Використовуйте конструкцію "as percent" для виведення відсотка активних користувачів. Приклад формату відповіді: 65.23).

Варіант 17

1. Виведіть інформацію про учнів, які живуть на вулиці Сумській (ul. Sumska)? (Поля в результуючій таблиці: *).

2. Скільки Ганн (Ganna) навчається в школі? (Поля в результуючій таблиці: count. Використовуйте конструкцію "as count" для агрегатної функції підрахунку кількості учнів. Це необхідно для коректної перевірки).

3. Скільки учнів у 10 В класі? (Поля в результуючій таблиці: count. Використовуйте конструкцію "as count" для агрегатної функції підрахунку кількості учнів. Це необхідно для коректної перевірки).

4. Який відсоток учнів навчається в "10 А" класі? Виведіть відповідь у діапазоні від 0 до 100 з округленням до чотирьох знаків після коми, наприклад, 96.0201 (Поля в результуючій таблиці: percent. Використовуйте конструкцію "as percent" для представлення результату обчислення. Це необхідно для коректної перевірки).

5. Який відсоток учнів народився у 2010 році? Результат округлити до цілого в менший бік. (Поля в результуючій таблиці: percent. Використовуйте конструкцію "as percent" для зазначення відсотка. Це необхідно для коректної перевірки).

6. Які кабінети найчастіше використовувалися для проведення занять? Виведіть ті, які використовувалися максимальну кількість разів. (Поля в результуючій таблиці: classroom).

Варіант 18

1. Визначити, хто з членів сім'ї купував картоплю (potato) (Поля в результуючій таблиці: status).

2. Знайти імена всіх матерів (mother) (Поля в результуючій таблиці: member_name).

3. Змініть ім'я "Andie Quincey" на нове "Andie Anthony".

4. Додайте товар з ім'ям "Cheese" і типом "food" до списку товарів (Goods). (В якості первинного ключа (good_id) вкажіть кількість записів у таблиці + 1).

5. Додайте до списку типів товарів (GoodTypes) новий тип "auto". (В якості первинного ключа (good_type_id) вкажіть кількість записів у таблиці + 1).

6. Видалити всіх членів сім'ї з прізвищем "Quincey".

Варіант 19

1. Виведіть ідентифікатор і ознаку наявності кухні в приміщенні. Якщо кухня у помешканні, що здається, присутня, то виведіть "YES", інакше "NO" (Поля в результуючій таблиці: id, has_kitchen. Використовуйте

конструкцію "AS has_kitchen" для виведення ознаки наявності кухні в приміщенні).

Формат очікуваного результату запиту:

	id	has_kitchen
1	1	NO
2	2	YES

2. Вивести користувачів, які вказали +353. (Поля в результуючій таблиці: *).

3. Вибрати всі записи, в яких рік верифікації email 2023 р.

4. Вивести імена всіх користувачів сервісу бронювання житла, а також дві ознаки: чи є користувач власником будь-якого житла (is_owner) і чи є користувач орендарем (is_tenant). У разі наявності у користувача ознаки необхідно вивести у відповідне поле 1, інакше 0 (Поля в результуючій таблиці: name, is_owner, is_tenant. Використовуйте конструкцію "AS is_owner" для відображення ознаки власника житла. Використовуйте конструкцію "AS is_tenant" для відображення ознаки орендаря).

5. Виведіть усіх користувачів з електронною поштою в «hotmail.com» (Поля в результуючій таблиці: *).

6. Додати відгук із рейтингом 5 на житло, що знаходяться за адресою "11218, Friel Place, New York", від імені "George Clooney" (В якості первинного ключа (id) вкажіть кількість записів у таблиці + 1. Резервування кімнати, на яку вам потрібно залишити відгук, уже було зроблено, потрібно лише її знайти).

Варіант 20

1. Виведіть інформацію про учнів, які живуть на вулиці Сумській (ul. Sumska)? (Поля в результуючій таблиці: *).

2. Скільки учнів у 10 В класі? (Поля в результуючій таблиці: count. Використовуйте конструкцію "as count" для агрегатної функції підсчета количества учащихся. Это необходимо для корректной проверки).

3. З'ясуйте, о котрій годині за розкладом починається четверте заняття. (Поля в результуючій таблиці: start_pair).

4. Перенести розклад усіх занять на 30 хв. уперед.

5. Виведіть відсортований список (за зростанням) прізвищ та імен студентів у вигляді Прізвище_Ім'я (Поля в результуючій таблиці: name).

6. Створіть подання з ім'ям "People", яке міститиме список імен (first_name) і прізвищ (last_name) усіх студентів (Student) і викладачів (Teacher).

7. Виведіть ідентифікатори викладачів, які хоча б один раз за весь час викладали в кожному з одинадцятих класів. (Поля в результуючій таблиці: teacher).

Варіант 21

1. Виведіть ідентифікатор і ознаку наявності телебачення в приміщенні. Якщо телебачення у помешканні, що здається, присутнє, то виведіть "YES", інакше "NO" (Поля в результуючій таблиці: id, has_tv. Використовуйте конструкцію "AS has_tv" для виведення ознаки наявності телебачення в приміщенні).

Формат очікуваного результату запиту:

	id	has_tv
1	1	NO
2	2	YES

2. Вивести користувачів, які вказали номер телефону +593 (Поля в результуючій таблиці: *).

3. Вибрати всі записи, в яких рік верифікації email 2022 р.

4. Виведіть поля id, home_type, price у всього житла з таблиці Rooms. Якщо кімната має телевізор та інтернет одночасно, то як ціну в полі price виведіть ціну, застосувавши знижку 10% (Поля в результуючій таблиці: id, home_type, price).

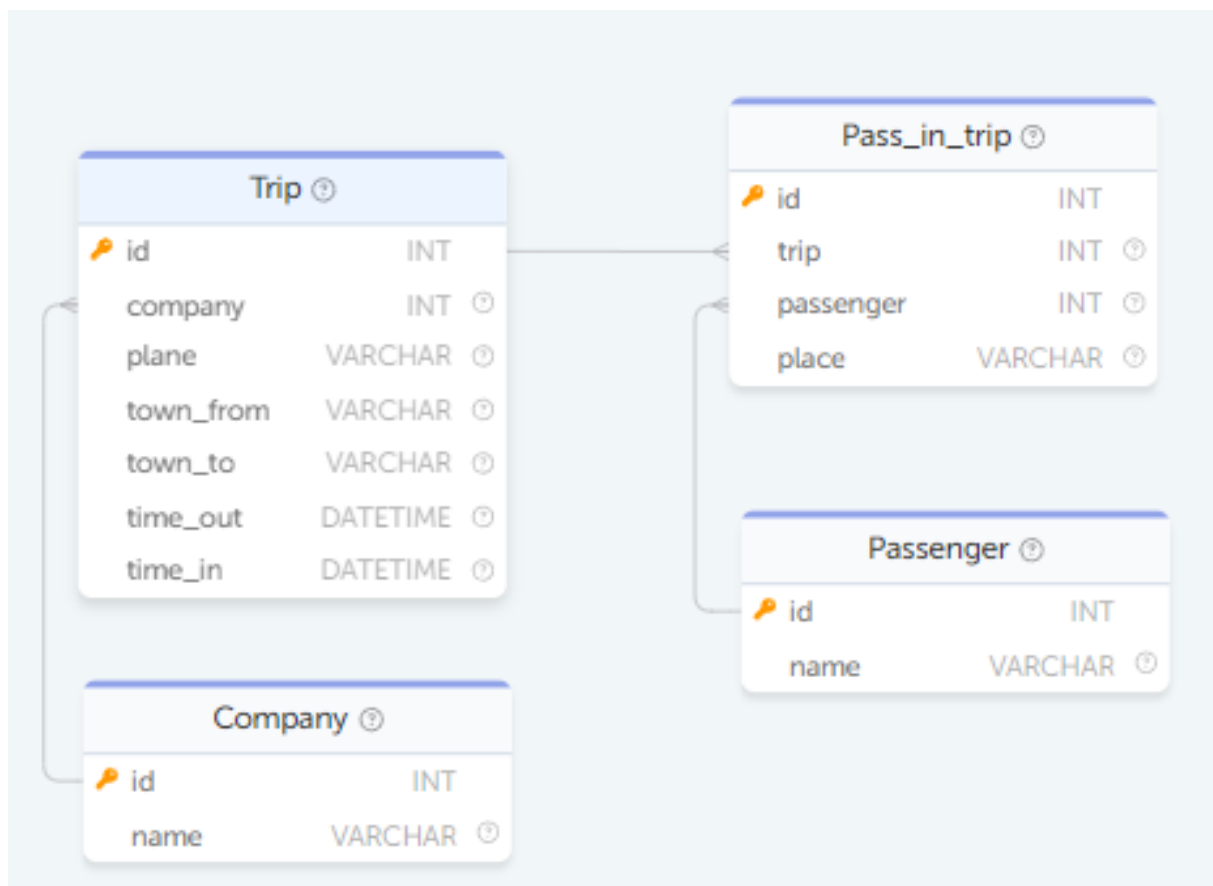
5. Створіть подання "Verified_Users" з полями id, name та email, яке показуватиме тільки тих користувачів, у яких підтверджена адреса електронної пошти.

6. Для кожної кімнати, яку винаймали щонайменше 1 раз, знайдіть ім'я людини, яка винаймала її востаннє, і дату, коли вона виїхала (Поля в результуючій таблиці: room_id, name, end_date. Використовуйте конструкцію "as room_id" для виведення ідентифікатора кімнати).

СХЕМИ ТА ТАБЛИЦІ БАЗ ДАНИХ ЗА ВАРІАНТАМИ

Варіанти 1, 5, 9, 13

Схема бази даних



Таблиці бази даних

Company

№	id	name
1	1	Мотор Січ
2	2	Дніпроавіа
3	3	Dale_avia
4	4	air_France
5	5	British_AW

Passenger

№	id	name
1	1	Bruce Willis
2	2	George Clooney
3	3	Kevin Costner
4	4	Donald Sutherland
5	5	Jennifer Lopez
6	6	Ray Liotta
7	7	Samuel L. Jackson
8	8	Nikole Kidman
9	9	Alan Rickman
10	10	Kurt Russell
11	11	Harrison Ford
12	12	Russell Crowe
13	13	Steve Martin
14	14	Michael Caine
15	15	Angelina Jolie
16	16	Mel Gibson
17	17	Michael Douglas
18	18	John Travolta
19	19	Sylvester Stallone
20	20	Tommy Lee Jones
21	21	Catherine Zeta-Jones
22	22	Antonio Banderas
23	23	Kim Basinger
24	24	Sam Neill
25	25	Gary Oldman
26	26	CLINT Eastwood
27	27	Brad Pitt
28	28	Johnny Depp
29	29	Pierce Brosnan
30	30	Sean Connery
31	31	Bruce Willis
32	37	Mullah Omar
33	38	Leonardo Grant-Baker

Pass_in_trip

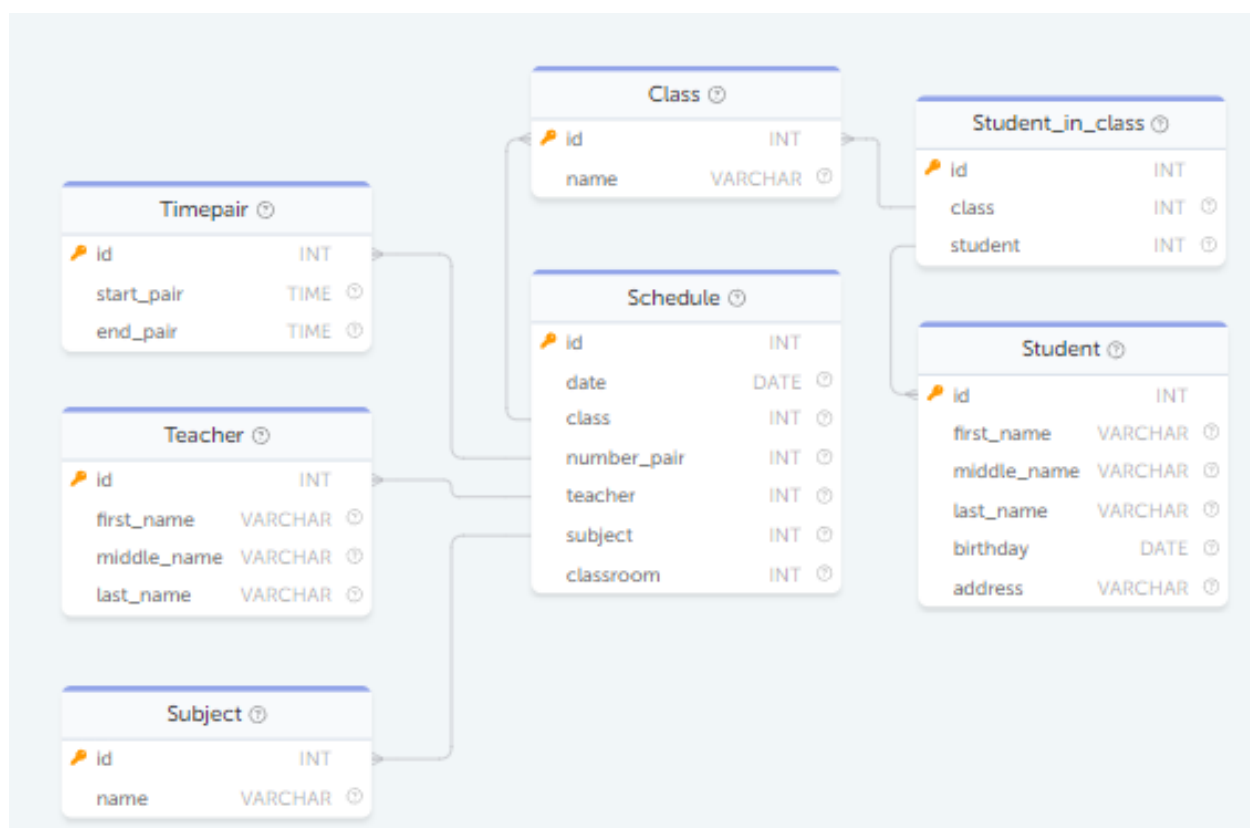
№	id	trip	passenger	place
1	1	1100	1	1a
2	2	1123	3	2a
3	3	1123	1	4c
4	4	1123	6	4b
5	5	1124	2	2d
6	6	1145	3	2c
7	7	1181	1	1a
8	8	1181	6	1b
9	9	1181	8	3c
10	10	1181	5	1b
11	11	1182	5	4b
12	12	1187	8	3a
13	13	1188	8	3a
14	14	1182	9	6d
15	15	1145	5	1d
16	16	1187	10	3d
17	17	8882	37	1a
18	18	7771	37	1c
19	19	7772	37	1a
20	20	8881	37	1d
21	21	7778	10	2a
22	22	7772	10	3a
23	23	7771	11	4a
24	24	7771	11	1b
25	25	7771	11	5a
26	26	7772	12	1d
27	27	7773	13	2d
28	28	7772	13	1b
29	29	8882	14	3d
30	30	7771	14	4d
31	31	7771	14	5d
32	32	7772	14	1c

Trip

№	id	company	plane	town_from	town_to	time_out	time_in
1	1100	4	Boeing	Lviv	Paris	2024-01-01T14:30:00.000Z	2024-01-01T17:50:00.000Z
2	1101	4	Boeing	Paris	Lviv	2024-01-01T08:12:00.000Z	2024-01-01T11:45:00.000Z
3	1123	3	Ембраер	Varshava	Kyiv	2024-01-01T16:20:00.000Z	2024-01-02T03:40:00.000Z
4	1124	3	Ембраер	Kyiv	Varshava	2024-01-01T09:00:00.000Z	2024-01-01T19:50:00.000Z
5	1145	2	Ербас А320-200	Brussels	Varshava	2024-01-01T09:35:00.000Z	2024-01-01T11:23:00.000Z
6	1146	2	Ербас А320-201	Varshava	Brussels	2024-01-01T17:55:00.000Z	2024-01-01T20:01:00.000Z
7	1181	1	Ан-12БП	Varshava	Brussels	2024-01-01T06:12:00.000Z	2024-01-01T08:01:00.000Z
8	1182	1	Ан-12БП	Brussels	Varshava	2024-01-01T12:35:00.000Z	2024-01-01T14:30:00.000Z
9	1187	1	Ан-12БП	Varshava	Brussels	2024-01-01T15:42:00.000Z	2024-01-01T17:39:00.000Z
10	1188	1	Ан-12БП	Brussels	Varshava	2024-01-01T22:50:00.000Z	2024-01-02T00:48:00.000Z
11	1195	1	Ембраер	Varshava	Brussels	2024-01-01T23:30:00.000Z	2024-01-02T01:11:00.000Z
12	1196	1	Ембраер	Brussels	Varshava	2024-01-01T04:00:00.000Z	2024-01-01T05:45:00.000Z
13	7771	5	Boeing	London	Singapore	2024-01-01T01:00:00.000Z	2024-01-01T11:00:00.000Z
14	7772	5	Boeing	Singapore	London	2024-01-01T12:00:00.000Z	2024-01-02T02:00:00.000Z
15	7773	5	Boeing	London	Singapore	2024-01-01T03:00:00.000Z	2024-01-01T13:00:00.000Z
16	7774	5	Boeing	Singapore	London	2024-01-01T14:00:00.000Z	2024-01-02T06:00:00.000Z
17	7775	5	Boeing	London	Singapore	2024-01-01T09:00:00.000Z	2024-01-01T20:00:00.000Z
18	7776	5	Boeing	Singapore	London	2024-01-01T18:00:00.000Z	2024-01-02T08:00:00.000Z
19	7777	5	Boeing	London	Singapore	2024-01-01T18:00:00.000Z	2024-01-02T06:00:00.000Z
20	7778	5	Boeing	Singapore	London	2024-01-01T22:00:00.000Z	2024-01-02T12:00:00.000Z
21	8881	5	Boeing	London	Paris	2024-01-01T03:00:00.000Z	2024-01-01T04:00:00.000Z
22	8882	5	Boeing	Paris	London	2024-01-01T22:00:00.000Z	2024-01-01T23:00:00.000Z

Варіанти 2, 6, 10, 14, 17, 20

Схема бази даних



Таблиці бази даних

Student

№	id	first_name	middle_name	last_name	birthday	address
1	1	Nikolaj	Fedorovych	Sokolov	2007-10-01 T00:00:00.000Z	ul. Sumska, d. 36, kv. 5
2	2	Vyacheslav	Yevgenovych	Eliseev	2007-11-21 T00:00:00.000Z	ul. Sumska, d. 40, kv. 7
3	3	Ivan	Antonovych	Efremov	2007-09-19 T00:00:00.000Z	ul. Sumska, d. 58, kv. 16
4	4	Anatolii	Valentynovych	ZHDanov	2007-07-15 T00:00:00.000Z	ul. Sumska, d. 21, kv. 7
5	5	Georgii	Dmytrovych	Noskov	2007-03-03 T00:00:00.000Z	ul. Sumska, d. 45, kv. 65
6	6	Artem	Borysovych	Sergeev	2007-01-01 T00:00:00.000Z	ul. Sumska, d. 1, kv. 5
7	7	Aryna	Fedorivna	Evseeva	2007-08-11 T00:00:00.000Z	ul. Sumska, d. 21, kv. 51
8	8	Angelina	Oleksandrivna	Voroncova	2007-05-21 T00:00:00.000Z	ul. Airocosmichna, d. 121, kv. 5
9	9	Kateryna	Oleksiiivna	Ustinova	2007-01-04 T00:00:00.000Z	ul. Aerocosmichna, d. 45, kv. 2
10	10	Rayisa	Semenivna	Lapina	2007-02-05 T00:00:00.000Z	ul. Aerocosmichna, d. 56, kv. 7

№	id	first_name	middle_name	last_name	birthday	address
11	11	Leonid	Mykytovych	Ignatov	2007-12-30 T00:00:00.000Z	ul. Sumska, d. 78, kv. 9
12	12	Snezhana	Yakivna	Seliverstova	2007-07-23 T00:00:00.000Z	ul. Sumska, d. 78, kv. 56
13	13	Semen	Vladyslavovych	Biryukov	2007-09-11 T00:00:00.000Z	ul. Ryzhivska, d. 54, kv. 67
14	14	Georgii	Vyacheslavovych	Baranov	2008-01-11 T00:00:00.000Z	ul. Ryzhivska, d. 56, kv. 89
15	15	Yuliya	Rostislavivna	Vishnyakova	2008-12-03 T00:00:00.000Z	ul. Manizera, d. 96, kv. 45
16	16	Valentyna	Andriivna	Bolshakova	2008-05-30 T00:00:00.000Z	ul. Aerocosmichna, d. 89, kv. 67
17	17	Leonid	Romanovych	Kryukov	2008-06-04 T00:00:00.000Z	ul. Gareta Djonsa, d. 78, kv. 9
18	18	Vladyslav	Sergiyovych	Cvetkov	2008-04-05 T00:00:00.000Z	ul. Turinska, d. 89, kv. 7
19	19	Snezhana	Ivanivna	Morozova	2008-11-06 T00:00:00.000Z	ul. Molochna, d. 7, kv. 9
20	20	Lyubov	Maksymivna	Borisova	2008-07-12 T00:00:00.000Z	ul. Pavlovska, d. 23, kv. 13
21	21	Anfisa	Antonivna	Kalashnikova	2008-01-11 T00:00:00.000Z	ul. Naukova, d. 4, kv. 4
22	22	Ganna	Olegivna	Osipova	2008-04-16 T00:00:00.000Z	ul. Ryzhivska, d. 32, kv. 56
23	23	Kristina	Fedorivna	Myasnikova	2008-02-03 T00:00:00.000Z	ul. Frankivska, d. 45, kv. 56
24	24	Kristina	Glibivna	Smirnova	2008-03-04 T00:00:00.000Z	ul. Yubileyana, d. 78, kv. 67
25	25	Borys	Innokentiyovych	Simonov	2008-04-05 T00:00:00.000Z	ul. Manizera, d. 89, kv. 78
26	26	Dmytro	Leonidovych	Trofimov	2008-05-06 T00:00:00.000Z	ul. Sumska, d. 78, kv. 9
27	27	Yakov	Leonidovych	Rozhkov	2008-06-07 T00:00:00.000Z	ul. Turinska, d. 45, kv. 7
28	28	Fedir	Rostislavovych	Drozdov	2008-07-08 T00:00:00.000Z	ul. Aerocosmichna, d. 23, kv. 34
29	29	Glib	Antonovych	Strelkov	2008-08-09 T00:00:00.000Z	ul. Gareta Djonsa, d. 23, kv. 13
30	30	Angelina	Oleksandrivna	Lukina	2008-09-10 T00:00:00.000Z	ul. Molochna, d. 45, kv. 7
31	31	Nina	Ilivna	Odincova	2008-10-11 T00:00:00.000Z	ul. Manizera, d. 4, kv. 76
32	32	Valeriya	Olegivna	Novikova	2008-11-12 T00:00:00.000Z	ul. Pavlovska, d. 3, kv. 56
33	33	Grygorii	Genadiyovych	Kapustin	2008-12-13 T00:00:00.000Z	ul. Aerocosmichna, d. 45, kv. 6
34	34	Vitalii	Eduardovych	Panfilov	2008-01-01 T00:00:00.000Z	ul. Ryzhivska, d. 34, kv. 87
35	35	Svyatoslav	Vyacheslavovych	Tarasov	2009-01-14 T00:00:00.000Z	ul. Sumska, d. 5, kv. 6
36	36	Matvey	Vyacheslavovych	YAkushev	2009-02-15 T00:00:00.000Z	ul. Gareta Djonsa, d. 7, kv. 8

№	id	first_name	middle_name	last_name	birthday	address
37	37	Illya	Stepanovych	Alekseev	2009-03-16 T00:00:00.000Z	ul. Aerocosmichna, d. 9, kv. 12
38	38	Lyubov	Vyacheslavivna	Zaharova	2009-04-17 T00:00:00.000Z	ul. Ryzhivska, d. 12, kv. 13
39	39	Polina	Kyrylivna	Sidorova	2009-05-18 T00:00:00.000Z	ul. Manizera, d. 13, kv. 14
40	40	Elyzaveta	Fedorivna	Samojlova	2009-06-19 T00:00:00.000Z	ul. Turinska, d. 46, kv. 45
41	41	Yuliya	Daniilivna	Avdeeva	2009-07-20 T00:00:00.000Z	ul. Molochna, d. 87, kv. 78
42	42	Matvey	Genadiyovych	Bogdanov	2009-08-21 T00:00:00.000Z	ul. Aerocosmichna, d. 9, kv. 8
43	43	Illya	Yegorovych	Filippov	2009-09-22 T00:00:00.000Z	ul. Ryzhivska, d. 100, kv. 90
44	44	Denys	Mykytovych	Mel	2009-10-23 T00:00:00.000Z	ul. Pavlovska, d. 123, kv. 45
45	45	Svyatoslav	Valentynovych	Muravyov	2009-11-24 T00:00:00.000Z	ul. Aerocosmichna, d. 45, kv. 67
46	46	Ganna	Denysivna	Kulagina	2009-12-25 T00:00:00.000Z	ul. Ryzhivska, d. 7, kv. 78
47	47	Zhanna	Illivna	Fokina	2009-01-26 T00:00:00.000Z	ul. Turinska, d. 8, kv. 9
48	48	Valeriya	Yuriiivna	Lapina	2009-02-27 T00:00:00.000Z	ul. Naukova, d. 78, kv. 56
49	49	Valentina	Andriivna	Sazonova	2009-03-01 T00:00:00.000Z	ul. Gareta Djonsa, d. 84, kv. 48
50	50	Nataliya	Igorivna	Myasnikova	2009-04-02 T00:00:00.000Z	ul. Sumska, d. 1, kv. 23
51	51	Viktoriya	Yaroslavivna	Makarova	2009-05-03 T00:00:00.000Z	ul. Pavlovska, d. 5, kv. 8
52	52	Stanislav	<NULL>	Lazarev	2009-06-04 T00:00:00.000Z	ul. Molochna, d. 6, kv. 9
53	53	Genadii	Denysovych	Ovchinnikov	2009-07-05 T00:00:00.000Z	ul. Manizera, d. 7, kv. 9
54	54	Roman	Mykolayovych	Shilov	2010-08-06 T00:00:00.000Z	ul. Sumska, d. 56, kv. 80
55	55	Tymur	Illich	Subbotin	2010-09-07 T00:00:00.000Z	ul. Gareta Djonsa, d. 47, kv. 39
56	56	Daniil	Ivanovych	Osipov	2010-10-08 T00:00:00.000Z	ul. Ryzhivska, d. 78, kv. 69
57	57	Arina	Tymofiivna	Silina	2010-11-09 T00:00:00.000Z	ul. Ryzhivska, d. 75, kv. 39
58	58	Nadiya	Illivna	Zaharova	2010-12-10 T00:00:00.000Z	ul. Molochna, d. 70, kv. 84
59	59	Larysa	Stanislavivna	Shcherbakova	2010-01-11 T00:00:00.000Z	ul. Aerocosmichna, d. 94, kv. 4
60	60	Oleksandra	Andriivna	Belozyorova	2010-02-12 T00:00:00.000Z	ul. Sumska, d. 3, kv. 21
61	61	Natalya	Mykolaivna	Davydova	2010-03-13 T00:00:00.000Z	ul. Gareta Djonsa, d. 4, kv. 2

№	id	first_name	middle_name	last_name	birthday	address
62	62	Mariya	Valeriivna	Fadeeva	2010-04-14 T00:00:00.000Z	ul. Ryzhivska, d. 8, kv. 5
63	63	Yurii	Denysovych	Markov	2010-05-15 T00:00:00.000Z	ul. Manizera, d. 9, kv. 5
64	64	Kyrylo	<NULL>	Shubin	2010-06-16 T00:00:00.000Z	ul. Gareta Djonsa, d. 6, kv. 3
65	65	Grygorii	Kyrylovych	Kolobov	2010-07-17 T00:00:00.000Z	ul. Ryzhivska, d. 9, kv. 34
66	66	Semen	Semenovych	Trofimov	2010-08-18 T00:00:00.000Z	ul. Turinska, d. 11, kv. 56
67	67	Vasyl	Genadiyovych	Ustinov	2010-09-17 T00:00:00.000Z	ul. Naukova, d. 11, kv. 13
68	68	Valentyna	Yakivna	SHarova	2010-10-04 T00:00:00.000Z	ul. Aerocosmichna, d. 31, kv. 3
69	69	Larysa	Tymurivna	Savina	2011-11-05 T00:00:00.000Z	ul. Gareta Djonsa, d. 22, kv. 33
70	70	Galyna	Yaroslavivna	Orehova	2011-12-07 T00:00:00.000Z	ul. Molochna, d. 77, kv. 87
71	71	Aryna	Ivanivna	Sharapova	2011-01-08 T00:00:00.000Z	ul. Pavlovska, d. 47, kv. 94
72	72	Viktoriya	Yuriivna	Sergeeva	2011-02-09 T00:00:00.000Z	ul. Manizera, d. 44, kv. 87
73	73	Vasyl	Glibovych	Krasilnikov	2011-03-15 T00:00:00.000Z	ul. Gareta Djonsa, d. 101, kv. 92
74	74	Tymur	Pylypovych	Rusakov	2011-04-26 T00:00:00.000Z	ul. Ryzhivska, d. 197, kv. 56
75	75	Glib	Tymofiyovych	Nesterov	2011-05-05 T00:00:00.000Z	ul. Aerocosmichna, d. 91, kv. 71
76	76	Denys	Matviyovych	Makarov	2011-05-09 T00:00:00.000Z	ul. Ryzhivska, d. 41, kv. 31
77	77	Elyzaveta	Borysivna	Shilova	2011-06-14 T00:00:00.000Z	ul. Turinska, d. 68, kv. 15
78	78	Vira	Lvivna	Evseeva	2011-07-04 T00:00:00.000Z	ul. Sumska, d. 5, kv. 13
79	79	Margaryta	Vladyslavivna	Kabanova	2011-08-08 T00:00:00.000Z	ul. Manizera, d. 98, kv. 67
80	80	Angelina	Danylivna	Lazareva	2011-09-04 T00:00:00.000Z	ul. Gareta Djonsa, d. 67, kv. 93
81	81	Semen	Anatoliyovych	Voronov	2011-10-05 T00:00:00.000Z	ul. Naukova, d. 89, kv. 89
82	82	Innokentii	Vyacheslavovych	Nekrasov	2011-01-06 T00:00:00.000Z	ul. Ryzhivska, d. 777, kv. 77
83	83	Artem	Viktorovych	Nikitin	2011-11-07 T00:00:00.000Z	ul. Molochna, d. 666, kv. 7
84	84	Yegor	Petrovych	Belyakov	2011-12-08 T00:00:00.000Z	ul. Gareta Djonsa, d. 87, kv. 9
85	85	Elyzaveta	<NULL>	Fadeeva	2008-11-01 T00:00:00.000Z	ul. Gareta Djonsa, d. 7, kv. 19

Schedule

№	id	date	class	number_pair	teacher	subject	classroom
1	1	2024-09-01 T00:00:00.000Z	9	1	11	1	47
2	2	2024-09-01 T00:00:00.000Z	9	2	8	2	13
3	3	2024-09-01 T00:00:00.000Z	9	3	4	3	13
4	4	2024-09-02 T00:00:00.000Z	9	1	4	3	13
5	5	2024-09-02 T00:00:00.000Z	9	2	2	4	34
6	6	2024-09-02 T00:00:00.000Z	9	3	6	5	35
7	7	2024-09-03 T00:00:00.000Z	9	1	5	6	36
8	8	2024-09-03 T00:00:00.000Z	9	2	13	7	37
9	9	2024-09-03 T00:00:00.000Z	9	3	6	8	38
10	10	2024-09-04 T00:00:00.000Z	9	1	9	9	39
11	11	2024-09-04 T00:00:00.000Z	9	2	10	10	40
12	12	2024-09-04 T00:00:00.000Z	9	3	3	11	41
13	13	2024-09-05 T00:00:00.000Z	9	1	3	13	43
14	14	2024-09-05 T00:00:00.000Z	9	2	11	1	47
15	15	2024-09-05 T00:00:00.000Z	9	3	5	6	36
16	16	2024-08-30 T00:00:00.000Z	9	1	2	4	34
17	17	2024-08-30 T00:00:00.000Z	9	2	8	2	13
18	18	2024-08-30 T00:00:00.000Z	9	3	6	5	35
19	19	2024-08-30 T00:00:00.000Z	9	4	10	1	47
20	20	2024-09-03 T00:00:00.000Z	9	4	10	10	40
21	21	2024-08-30 T00:00:00.000Z	8	1	7	9	53
22	22	2024-08-30 T00:00:00.000Z	8	2	7	9	53
23	23	2024-08-30 T00:00:00.000Z	8	3	8	2	38
24	24	2024-08-30 T00:00:00.000Z	8	4	11	1	43
25	25	2024-08-30	8	5	8	3	39

№	id	date	class	number_pair	teacher	subject	classroom
		T00:00:00.000Z					
26	26	2024-09-01 T00:00:00.000Z	8	2	2	4	34
27	27	2024-09-01 T00:00:00.000Z	8	3	6	5	35
28	28	2024-09-01 T00:00:00.000Z	8	4	12	6	36
29	29	2024-09-01 T00:00:00.000Z	8	5	13	7	37
30	30	2024-09-02 T00:00:00.000Z	8	3	6	8	38
31	31	2024-09-02 T00:00:00.000Z	8	4	7	9	53
32	32	2024-09-03 T00:00:00.000Z	8	1	10	10	40
33	33	2024-09-03 T00:00:00.000Z	8	2	7	9	53
34	34	2024-09-03 T00:00:00.000Z	8	3	7	9	53
35	35	2024-09-04 T00:00:00.000Z	8	1	1	11	4
36	36	2024-09-04 T00:00:00.000Z	8	2	1	12	42
37	37	2024-09-04 T00:00:00.000Z	8	3	3	13	43
38	38	2024-09-04 T00:00:00.000Z	8	4	8	2	42
39	39	2024-09-04 T00:00:00.000Z	8	5	11	1	43
40	40	2024-09-05 T00:00:00.000Z	8	2	11	1	43
41	41	2024-09-03 T00:00:00.000Z	7	1	8	2	42
42	42	2024-09-03 T00:00:00.000Z	7	2	4	3	13
43	43	2024-09-03 T00:00:00.000Z	7	3	15	10	40
44	44	2024-09-02 T00:00:00.000Z	7	2	6	5	35

Studen_in_class

№	id	class	student
1	67	9	1
2	68	9	2
3	69	9	3
4	57	9	4
5	71	9	5
6	72	9	6
7	73	9	7
8	74	9	8
9	75	9	9
10	76	9	10
11	77	9	11
12	78	9	12
13	79	9	13
14	80	8	14
15	81	8	15
16	82	8	16
17	83	8	17
18	84	8	18
19	85	8	19

Subject

№	id	name
1	1	Math
2	2	Ukrainian language
3	3	Ukrainian Literature
4	4	Physics
5	5	Chemistry
6	6	Geography
7	7	History
8	8	Biology
9	9	English language
10	10	Informatics
11	11	Physical Culture
12	12	Robotics
13	13	Technology

Class

№	id	name
1	1	8 A
2	2	8 B
3	3	9 C
4	4	9 B
5	5	9 A
6	6	10 B
7	7	10 A
8	8	11 B
9	9	11 A
10	10	7 A
11	11	7 B
12	13	6 A
13	14	6 B
14	15	5 A
15	16	5 B
16	17	4 A

Teacher

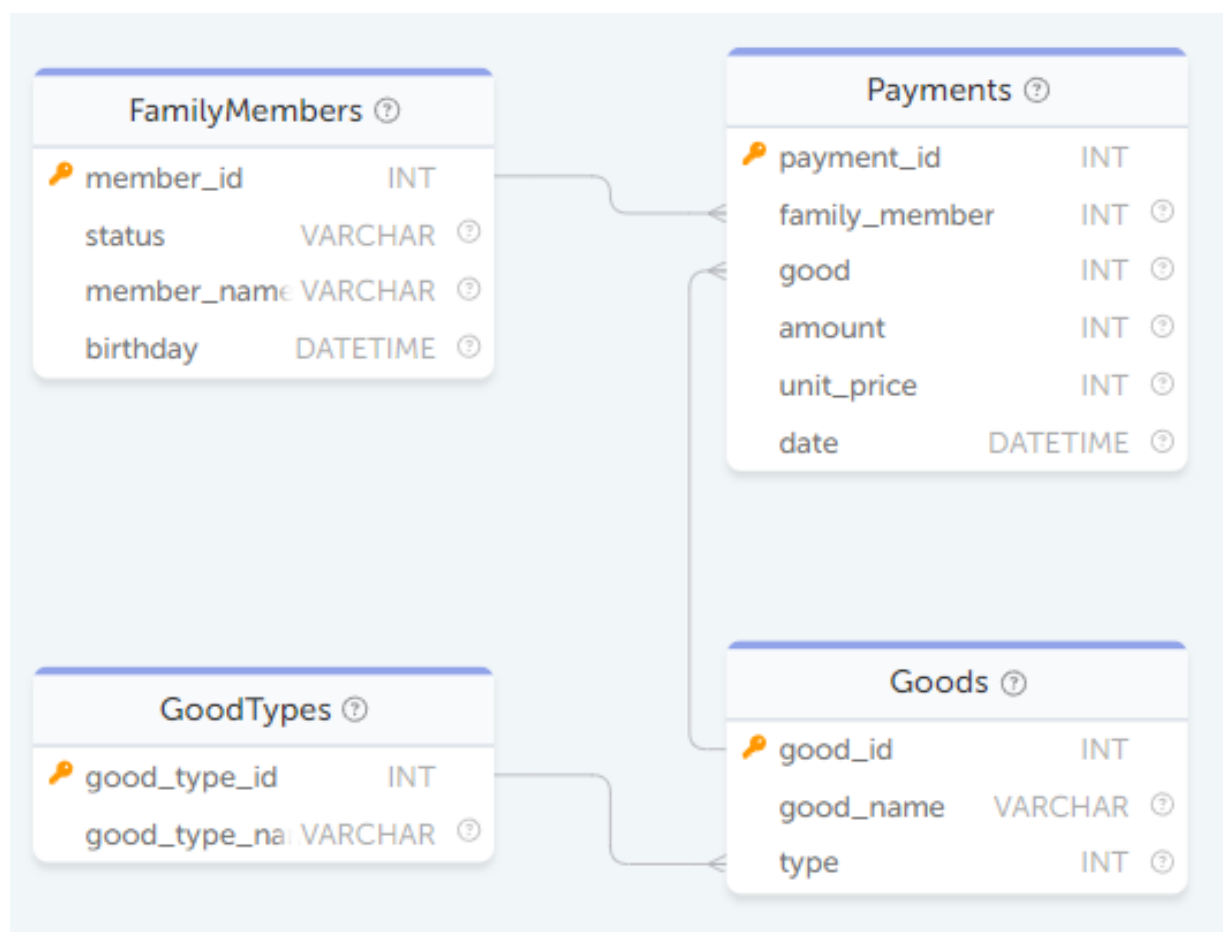
№	id	first_name	middle_name	last_name
1	1	Pavlo	Petrovych	Romashkin
2	2	Vasyl	Antonovych	Kolesnikov
3	3	Mariya	Stepanovna	Vaulina
4	4	Tayisiya	Tymurovna	Zhedrinska
5	5	Maryna	Genadiivna	Sosnovska
6	6	Kateryna	Yakovlevna	Myasoedova
7	7	Liliya	Tymurovna	Arakcheeva
8	8	Sofia	Dmitrivna	Krauze
9	9	Viktoriya	Mykolaivna	Ostrozhska
10	10	Yurij	<NULL>	Krylov
11	11	Andrii	<NULL>	Evseev
12	12	Bogdan	Romanovych	Moiseev
13	13	Oleksandr	Volodymyrovych	Sobolev
14	14	Vasyl	Vasyliovych	Vasylenko
15	15	Petro	Oleksiyovych	Romashkin

Timepair

№	id	start_pair	end_pair
1	1	8:30:00	9:15:00
2	2	9:20:00	10:05:00
3	3	10:15:00	11:00:00
4	4	11:05:00	11:50:00
5	5	12:50:00	13:35:00
6	6	13:40:00	14:25:00
7	7	14:35:00	15:20:00
8	8	15:25:00	16:10:00

Варіанти 3, 7, 11, 15, 18

Схема бази даних



Таблиці бази даних

FamilyMembers

№	member_id	status	member_name	birthday
1	1	father	Headley Quincey	1960-05-13 T00:00:00.000Z
2	2	mother	Flavia Quincey	1963-02-16 T00:00:00.000Z
3	3	son	Andie Quincey	1983-06-05 T00:00:00.000Z
4	4	daughter	Lela Quincey	1985-06-07 T00:00:00.000Z
5	5	daughter	Annie Quincey	1988-04-10 T00:00:00.000Z
6	6	father	Ernest Forrest	1961-09-11 T00:00:00.000Z
7	7	mother	Constance Forrest	1968-09-06 T00:00:00.000Z
8	8	daughter	Wednesday Addams	2005-01-13 T00:00:00.000Z

Payments

№	payment_id	date	family_member	good	amount	unit_price
1	1	2023-02-12 T00:00:00.000Z	1	1	1	2000
2	2	2023-03-23 T00:00:00.000Z	2	1	1	2100
3	3	2023-05-14 T00:00:00.000Z	3	4	5	20
4	4	2023-07-22 T00:00:00.000Z	4	5	1	350
5	5	2023-07-26 T00:00:00.000Z	4	7	2	150
6	6	2023-02-20 T00:00:00.000Z	5	6	1	100
7	7	2023-07-30 T00:00:00.000Z	2	6	1	120
8	8	2023-09-12 T00:00:00.000Z	2	16	1	5500
9	9	2023-09-30 T00:00:00.000Z	5	15	1	230
10	10	2023-10-27 T00:00:00.000Z	5	15	1	230
11	11	2023-11-28 T00:00:00.000Z	5	15	1	250
12	12	2023-12-22 T00:00:00.000Z	5	15	1	250
13	13	2023-08-11 T00:00:00.000Z	3	13	1	2200
14	14	2023-10-23 T00:00:00.000Z	2	14	1	66000
15	15	2023-02-03 T00:00:00.000Z	1	9	5	8
16	16	2023-03-11 T00:00:00.000Z	1	9	5	7
17	17	2023-03-18 T00:00:00.000Z	2	9	3	8
18	18	2023-04-20 T00:00:00.000Z	1	9	8	8
19	19	2023-05-13 T00:00:00.000Z	1	9	5	7
20	20	2023-06-11 T00:00:00.000Z	2	9	3	150
21	21	2024-01-12 T00:00:00.000Z	3	10	1	100
22	22	2024-03-12 T00:00:00.000Z	1	5	3	10
23	23	2023-06-05 T00:00:00.000Z	1	8	1	300
24	24	2023-06-20 T00:00:00.000Z	3	6	8	150
25	25	2023-06-21 T00:00:00.000Z	2	9	3	150
26	26	2023-06-22 T00:00:00.000Z	5	18	1	16
27	27	2023-06-22	3	17	1	59

№	payment_id	date	family_member	good	amount	unit_price
		T00:00:00.000Z				

Goods

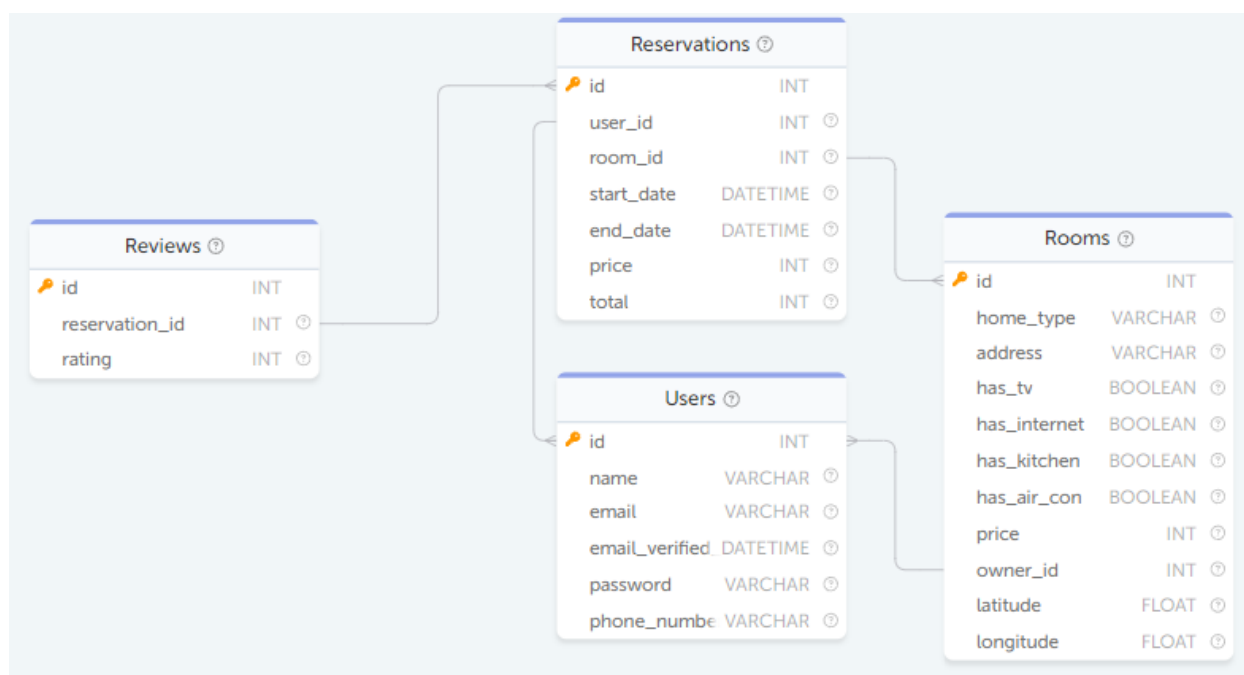
№	good_id	good_name	type
1	1	apartment fee	1
2	2	phone fee	1
3	3	bread	2
4	4	milk	2
5	5	red caviar	3
6	6	cinema	4
7	7	black caviar	3
8	8	cough tablets	5
9	9	potato	2
10	10	pineapples	3
11	11	television	8
12	12	vacuum cleaner	8
13	13	jacket	7
14	14	fur coat	7
15	15	music school fee	6
16	16	english school fee	6
17	17	airsoft game session	4
18	18	netflix subscription	4

GoodTypes

№	good_type_id	good_type_name
1	1	communal payments
2	2	food
3	3	delicacies
4	4	entertainment
5	5	treatment
6	6	education
7	7	clothes
8	8	equipment

Варіанти 4, 8, 12, 16, 19, 21

Схема бази даних



Таблиці бази даних

Users

№	id	name	email	email_verified_at	password	phone_number
1	1	Bruce Willis	barjam@hotmail.com	<NULL>	o8sdfsmgtswxjvxxl31	+593 123456789
2	2	George Clooney	tellis@me.com	2021-02-22 T00:17:42.000Z	sx8pav3ggvfwdz1yte7	+593 123456790
3	3	Kevin Costner	metzzo@superhotmail.com	2021-03-26 T00:01:52.000Z	l51q47urv4hbqqa5f87g	+593 123456791
4	4	Donald Sutherland	ralamosm@verizon.net	2021-11-03 T13:01:34.000Z	pj2625765oehrpk6lo4	+593 123456792
5	5	Jennifer Lopez	barjam@hotmail.com	2021-03-02 T09:35:57.000Z	c70n1nz39ggc9881ybst	+593 123456793
6	6	Ray Liotta	jdhedden@comcast.net	2022-02-24 T04:08:45.000Z	qpnuvi7mndgl1qmxgdl1	+593 123456794
7	7	Samuel L. Jackson	moonlapse@outlook.com	2023-07-19 T11:16:13.000Z	i6yvht95527z3idgqx9y	+593 123456795
8	8	Nikole Kidman	okroeger@msn.com	2021-09-17 T14:47:01.000Z	3bhg1l720u1reugsmbfc	+593 123456796
9	9	Alan Rickman	karasik@verizon.net	2021-10-03 T14:47:09.000Z	86ezqmymrhof0w81x3n9	+593 123456797
10	10	Kurt Russell	gator@live.com	2021-08-18 T22:26:00.000Z	4wwzx2kpn81410q79ieh	+593 123456798
11	11	Harrison Ford	kostas@hotmail.com	2021-12-05 T07:35:14.000Z	u8sxb8byhaajee65n9t2	+593 123456799
12	12	Russell Crowe	glenz@att.net	2023-01-04 T21:20:27.000Z	ywtkc8tns9le6aioalyf	+593 123456800
13	13	Steve Martin	nelson@outlook.com	2021-07-29 T04:25:00.000Z	w76yphg3kvzg77ilmxfs	+593 123456801
14	14	Michael Caine	dmouse@live.com	2021-04-24 T01:31:46.000Z	7k6m698ky9de34yc9zrh	+593 123456802

№	id	name	email	email_verified_at	password	phone_number
15	15	Angelina Jolie	chance@att.net	<NULL>	f403jj1cpkzoiwcn0kz	+593 123456803
16	16	Mel Gibson	rohitm@yahoo.ca	2021-01-02 T11:42:58.000Z	hojtsy7rt0v7we42vlr	+593 123456804
17	17	Michael Douglas	timtroyr@hotmail.com	2021-08-02 T12:50:51.000Z	03ga9r324u8n25ujb5ue	+593 123456805
18	18	John Travolta	wainwrig@msn.com	2021-11-19 T12:30:43.000Z	fzjhl0v82o0amaln8649	+593 123456806
19	19	Sylvester Stallone	kohlis@aol.com	2023-10-04 T00:16:57.000Z	b26226km8w1qfzqi9so	+353 209106141
20	20	Tommy Lee Jones	szymansk@live.com	2023-03-09 T11:08:34.000Z	c46s9eodm9k3qiwwju3y	+353 209122322
21	21	Catherine Zeta-Jones	flakeg@hotmail.com	2021-04-02 T12:28:19.000Z	izq356fwuzdhc6u1skan	+353 209113955
22	22	Antonio Banderas	sassen@verizon.net	2023-03-23 T08:15:47.000Z	wftjnkjfqj7dj7g6oeo	+353 209179717
23	23	Kim Basinger	jaesenj@sbcglobal.net	2022-09-27 T00:28:22.000Z	h60m7wovjopzp9kx4m3 m	+353 209133636
24	24	Sam Neill	clkao@live.com	2022-02-05 T22:24:13.000Z	3rpgiccs25fj7apg246s	+353 209141612
25	25	Hideo Kojima	admin@ sql-academy.org	2022-03-08 T14:15:51.000Z	97shpolky5vg3b2qn2wu	+61 1234567890
26	26	Clint Eastwood	froodian@verizon.net	2024-02-10 T09:43:59.000Z	ogbjpnp7vlzwxu4bj0bs	+43 1234567890
27	27	Brad Pitt	kewley@optonline.net	2022-02-11 T05:45:15.000Z	829j2ygocn8btzae49kv	+994 123456789
28	28	Johnny Depp	cgarcia@yahoo.ca	2022-05-26 T01:19:06.000Z	qpp6hbnae42cdhmxlk4j	+213 123456789
29	29	Pierce Brosnan	treeves@icloud.com	2024-03-08 T01:56:00.000Z	lqiweccne9rv8woo2go	+1907 12345678
30	30	Sean Connery	jschauma@icloud.com	2021-05-21 T00:45:17.000Z	lyh4jkdskvtvulvqi5db	+1264 12345678
31	31	Bruce Willis	kewley@icloud.com	2021-12-08 T20:18:59.000Z	0ofa2khvnptiackbssv0	+375 154771
32	32	Mullah Omar	jgoerzen@me.com	2024-07-18 T06:50:42.000Z	f3ft4fwo7vafvqb0ugk5	+375 154802
33	33	Vasanta Roberta	roberta@me.com	2022-08-11 T06:43:43.000Z	f3ftsdfsdfvqb0ugk5	+1441 12345678

Reservation

№	id	user_id	room_id	start_date	end_date	price	total
1	1	1	1	2021-11-13 T12:00:00.000Z	2021-11-15 T12:00:00.000Z	140	280
2	2	2	1	2021-10-09 T12:00:00.000Z	2021-10-12 T12:00:00.000Z	120	360
3	3	12	1	2022-12-29 T12:00:00.000Z	2023-01-07 T12:00:00.000Z	180	1620
4	4	13	1	2023-02-01 T12:00:00.000Z	2023-02-02 T12:00:00.000Z	145	145
5	5	17	1	2023-02-02 T12:00:00.000Z	2023-02-04 T12:00:00.000Z	149	298
6	6	14	2	2024-03-21 T09:00:00.000Z	2024-03-23 T09:00:00.000Z	225	450
7	7	19	13	2024-03-21 T10:00:00.000Z	2024-04-21 T10:00:00.000Z	89	2679
8	8	29	16	2023-06-14 T10:00:00.000Z	2023-06-24 T10:00:00.000Z	140	1400
9	9	32	21	2023-05-11	2023-05-14	270	810

№	id	user_id	room_id	start_date	end_date	price	total
				T10:00:00.000Z	T10:00:00.000Z		
10	10	13	21	2023-11-04 T10:00:00.000Z	2023-11-06 T10:00:00.000Z	280	560
11	11	9	21	2024-02-28 T10:00:00.000Z	2024-02-29 T10:00:00.000Z	290	290
12	12	7	19	2024-05-01 T10:00:00.000Z	2024-05-02 T10:00:00.000Z	99	99
13	13	15	8	2024-01-19 T12:00:00.000Z	2024-01-21 T12:00:00.000Z	78	156
14	14	13	7	2023-09-13 T10:00:00.000Z	2023-09-17 T10:00:00.000Z	60	240
15	15	12	5	2024-05-14 T10:00:00.000Z	2024-05-15 T10:00:00.000Z	80	80
16	16	10	5	2024-03-26 T10:00:00.000Z	2024-03-27 T10:00:00.000Z	80	80
17	17	11	50	2023-11-18 T11:00:00.000Z	2023-11-25 T11:00:00.000Z	80	560
18	18	27	49	2023-09-18 T11:00:00.000Z	2023-09-20 T11:00:00.000Z	110	220
19	19	28	49	2024-04-01 T11:00:00.000Z	2024-04-02 T11:00:00.000Z	115	115
20	20	1	49	2024-06-01 T10:00:00.000Z	2024-06-11 T10:00:00.000Z	115	1150
21	21	2	48	2023-11-07 T10:00:00.000Z	2023-11-10 T10:00:00.000Z	110	330
22	22	12	32	2024-01-14 T13:00:00.000Z	2024-01-18 T13:00:00.000Z	52	208
23	23	12	32	2023-09-17 T13:00:00.000Z	2023-09-18 T13:00:00.000Z	52	52
24	24	7	19	2024-01-08 T13:00:00.000Z	2024-01-11 T13:00:00.000Z	99	297
25	25	18	19	2023-11-01 T10:00:00.000Z	2023-11-16 T10:00:00.000Z	99	1485
26	26	21	17	2023-11-03 T09:00:00.000Z	2023-11-05 T09:00:00.000Z	215	430
27	27	31	25	2024-04-20 T09:00:00.000Z	2024-04-22 T09:00:00.000Z	120	240
28	28	21	14	2024-02-08 T10:00:00.000Z	2024-02-12 T10:00:00.000Z	85	340
29	29	21	39	2023-12-08 T10:00:00.000Z	2023-12-09 T10:00:00.000Z	150	150
30	30	3	38	2024-03-20 T10:00:00.000Z	2024-03-23 T10:00:00.000Z	185	555

Rooms

№	id	home_type	address	has_tv	has_internet	has_kitchen	has_air_con	price	owner_id	latitude	longitude
1	1	Private room	11218, Friel Place, New York	1	1	1	0	149	1	40.6475	-73.9724
2	2	Entire home/apt	10018, 6th Avenue, New York	0	1	1	1	225	1	40.7536	-73.9838
3	3	Private room	10027, West 128th Street, New York	1	0	0	1	150	2	40.809	-73.9419
4	4	Entire home/apt	11238, Gates Avenue, New York	1	0	0	1	89	2	40.6851	-73.9598
5	5	Entire	10029, Park	0	1	1	1	80	3	40.7985	-73.944

№	id	home_type	address	has_tv	has_internet	has_kitchen	has_air_con	price	owner_id	latitude	longitude
		home/apt	Avenue, New York								
6	6	Entire home/apt	10016, East 38th Street, New York	0	1	0	0	200	3	40.7477	-73.975
7	7	Private room	11216, Lexington Avenue, New York	0	0	1	1	60	4	40.6869	-73.956
8	8	Private room	10019, West 53rd Street, New York	1	0	0	1	79	14	40.7649	-73.9849
9	9	Private room	10025, West 107th Street, New York	1	0	1	0	79	6	40.8018	-73.9672
10	10	Entire home/apt	10002, Rutgers Street, New York	1	0	1	0	150	7	40.7134	-73.9904
11	11	Entire home/apt	10025, West 109th Street, New York	1	1	0	0	135	8	40.8032	-73.9654
12	12	Private room	10036, West 47th Street, New York	0	0	1	1	85	10	40.7608	-73.9887
13	13	Private room	11215, 11th Street, New York	0	0	0	0	89	11	40.6683	-73.9878
14	14	Private room	10025, West 106th Street, New York	0	0	0	1	85	12	40.7983	-73.9611
15	15	Entire home/apt	10014, Perry Street, New York	1	0	0	1	120	13	40.7353	-74.0052
16	16	Entire home/apt	11211, South 4th Street, New York	0	0	1	0	140	14	40.7084	-73.9535
17	17	Entire home/apt	11205, Willoughby Avenue, New York	0	0	1	0	215	15	40.6917	-73.9718
18	18	Private room	10011, West 21st Street, New York	1	1	0	1	140	16	40.7419	-73.995
19	19	Entire home/apt	11216, Bergen Street, New York	1	1	1	1	99	17	40.6759	-73.9469
20	20	Entire home/apt	10026-2864, Duke Ellington Circle, New York	0	0	0	0	190	18	40.7968	-73.9487
21	21	Entire home/apt	11249, North 8th Street, New York	1	1	1	1	299	19	40.7184	-73.9572
22	22	Private room	11217, Saint Marks Avenue, New York	1	0	0	1	130	20	40.6807	-73.9771
23	23	Private room	11217, 5th Avenue, New York	1	1	1	1	80	21	40.6799	-73.978
24	24	Private room	11217, 5th Avenue, New York	1	1	1	0	110	22	40.68	-73.9787
25	25	Entire	11221,	1	0	0	0	120	23	40.6837	-73.9403

№	id	home_type	address	has_tv	has_internet	has_kitchen	has_air_con	price	owner_id	latitude	longitude
		home/apt	Hancock Street, New York								
26	26	Private room	11218, Terrace Place, New York	1	1	0	0	60	24	40.656	-73.9752
27	27	Private room	10034, Seaman Avenue, New York	1	1	0	0	80	25	40.8675	-73.9264
28	28	Entire home/apt	10019, West 56th Street, New York	1	1	1	0	150	26	40.7672	-73.9853
29	29	Private room	10034, Post Avenue, New York	1	0	0	0	44	27	40.8648	-73.9211
30	30	Entire home/apt	10003, East 10th Street, New York	0	1	1	0	180	28	40.7292	-73.9854
31	31	Private room	10031, West 140th Street, New York	0	0	1	1	50	29	40.8224	-73.951
32	32	Private room	10027, West 126th Street, New York	0	0	1	1	52	30	40.813	-73.9547
33	33	Private room	11222, Lombardy Street, New York	0	1	0	0	55	31	40.7222	-73.9376
34	34	Private room	10031, West 138th Street, New York	1	0	1	0	50	32	40.8213	-73.9532
35	35	Private room	11216, Jefferson Avenue, New York	1	0	0	0	70	32	40.6831	-73.9547
36	36	Private room	11215, 10th Street, New York	1	0	0	1	89	32	40.6687	-73.9878
37	37	Private room	11221, Lexington Avenue, New York	0	1	0	0	35	2	40.6888	-73.9431
38	38	Entire home/apt	11237, Troutman Street, New York	1	0	1	0	85	1	40.7019	-73.9275
39	39	Private room	11226, Marlborough Road, New York	0	1	0	0	150	25	40.637	-73.9633
40	40	Shared room	10002, East Broadway, New York	1	1	1	1	40	25	40.714	-73.9892
41	41	Private room	10009, East 4th Street, New York	0	0	0	0	68	25	40.7229	-73.982
42	42	Entire home/apt	11215, 13th Street, New York	1	1	1	0	120	25	40.6628	-73.9797
43	43	Private room	11205, Park Avenue, New York	1	0	1	1	120	12	40.6967	-73.9758
44	44	Private room	10024, West 87th Street, New York	0	1	1	0	135	21	40.7901	-73.9793

№	id	home_type	address	has_tv	has_internet	has_kitchen	has_air_con	price	owner_id	latitude	longitude
45	45	Entire home/apt	10027, West 130th Street, New York	0	1	1	0	150	21	40.8117	-73.9448
46	46	Entire home/apt	11225, Ocean Avenue, New York	0	0	1	1	150	21	40.6594	-73.9624
47	47	Private room	11101, 21st Street, New York	1	0	0	1	130	26	40.7477	-73.9474
48	48	Entire home/apt	11207, Fulton Street, New York	1	1	0	0	110	16	40.6811	-73.9559
49	49	Entire home/apt	11221, Throop Avenue, New York	1	1	0	0	115	15	40.6855	-73.9409
50	50	Private room	11205, Washington Park, New York	1	0	1	1	80	15	40.6914	-73.9738

Reviews

№	id	reservation_id	rating
1	1	1	4
2	2	3	5
3	3	4	5
4	4	5	5
5	5	8	4
6	6	9	3
7	7	10	2
8	8	12	2
9	9	13	5
10	10	14	4
11	11	15	3
12	12	16	2
13	13	17	5
14	14	18	4
15	15	19	4
16	16	20	4
17	17	21	5
18	18	25	5
19	19	26	4
20	20	28	1
21	21	29	2
22	22	30	3

РЕКОМЕНДОВАНА ЛІТЕРАТУРА

Базова

1. Балик Н. Р., Мандзюк В. І. Бази даних MySQL: навч. посібн. – Тернопіль: навч. книга – Богдан, 2010. – 160 с.
2. Булатецька Л. В. Мова запитів SQL: текст лекцій нормативної навчальної дисципліни “Бази даних та розподілені інформаційно-аналітичні системи” / Булатецька Леся Віталіївна, Булатецький Віталій Вікторович. – Луцьк: СХУ імені Лесі Українки, 2018. – 92 с.
3. Гайдаржи В. І., Ізварін І. В. Бази даних в інформаційних системах. – Видавництво: Університет “Україна”, 2018. – С. 418.
4. Табличний процесор MS EXCEL: просунутий рівень. Практикум [Електронний ресурс] / уклад. А. А. Гаврилова, Н. О. Бринза, О. Г. Король; Харківський національний економічний університет ім. С. Кузнеця. – Електрон. текстові дан. (7,92 МБ). – Харків : ХНЕУ ім. С. Кузнеця, 2021. – 242 с.
5. Харів Н. О. Бази даних та інформаційні системи: навч. посібн. – Рівне: НУВГП, 2018. – 127 с.
6. Borrie H. The Firebird Book Second Edition // Volume 2: Developing with Firebird Data. – CreateSpace Independent Publishing Platform, 2013. – С. 532.
7. Borrie H. The Firebird Book Second Edition // Volume 3: Administering Firebird Servers and Database. – CreateSpace Independent Publishing Platform, 2013. – С. 444.
8. Maymala J. PostgreSQL for Data Architects. – Birmingham: Published by Packt Publishing Ltd., 2015. – 272 p.

Допоміжна

9. Каштан В.Ю., Іванов Д.В. Конспект лекцій з дисципліни “Бази даних в інформаційних системах”. Для студентів галузі знань 12 “Інформаційні технології” спеціальності 126 “Інформаційні системи та технології”. – Д.: НТУ “ДП”, 2021. – 58 с.
10. Мулеса О.Ю. Інформаційні системи та реляційні бази даних. навч. посібн. – Електронне видання, 2018. – 118 с.
11. Організація баз даних та знань. Проектування баз даних: навчально-практичний посібник для самостійної підготовки студентів. Ч. 1 / О. В. Тарасов, В. В. Федько, М. Ю. Лосєв. – Х.: Вид. ХНЕУ, 2011. – 200 с.

12. Ялова К. М., Якова К. В., Кондратенко А. І. Технологія OLAP для аналізу реляційних баз даних // Міжнародний мультидисциплінарний науковий журнал “ΛΟΓΟΣ. Мистецтво наукової думки”. – №8. – 2019. – С. 69 – 73.

13. Murach J. Murach’s MySQL (3rd Edition). – Publisher: Mike Murach & Associates, Inc., 2019. – 628 p.

Інформаційні ресурси

14. Встановлення MySQL – [Електронний ресурс]. – Режим доступу: <https://www.mysql.com/products>.

15. Встановлення pgAdmin – [Електронний ресурс]. – Режим доступу: <https://www.youtube.com/watch?v=kj-6qQL49c>.

16. Ковтун В., Белясник Л. Управління ключами та сертифікатами для систем автоматизації банку. – [Електронний ресурс]. – Режим доступу: https://www.cipher.com.ua/media/%D0%9D%D0%BE%D0%B2%D0%BE%D1%81%D1%82%D0%B8/2016/PKI-FORUM%202016/Presentation_Firebird_Crypto.pdf.

17. Средства для СУБД Firebird. – [Електронний ресурс]. – Режим доступу: <https://cipher.com.ua/de/products/facilities-firebird>.

18. Тренажер SQL – [Електронний ресурс]. – Режим доступу: https://www.w3schools.com/sql/sql_exercises.asp.

19. Firebird. – [Електронний ресурс]. – Режим доступу: <https://firebirdsql.org/en/firebird-3-0-10>.

Навчальне видання

ГАВРИЛОВА Алла Андріївна
ПОГАСІЙ Сергій Сергійович
КОРОЛЬОВ Роман Володимирович
ХВОСТЕНКО Владислав Сергійович
МІЛЕВСЬКА Тетяна Сергіївна

ТЕХНОЛОГІЇ БАЗ ДАНИХ

Навчально-практичний посібник

Серія «Кібербезпека та штучний інтелект»

За загальною редакцією доктора технічних наук, професора С. П. Євсєєва

Відповідальний за випуск *С. П. Євсєєв*

Керівник видавничих проектів *С. В. Піча*
Дизайн та верстка *К. А. Рижова*

Підписано до друку 08.02.2025 р. Зам. № 2025-10.

Формат 60×84/16. Гарнітура Times New Roman. Папір офсетний.

Обл.-вид. арк. 13,75. Ум.-друк. арк. 11,1.

Видавництво ПП «Новий Світ-2000»

e-mail: novsv2000@gmail.com

Свідоцтво про внесення суб'єкта видавничої справи до Державного реєстру
видавців і розповсюджувачів видавничої продукції: серія ДК № 59 від
25.05.2000 року, видане Державним комітетом інформаційної політики,
телебачення та радіомовлення України.

Виготовлено : Видавець ФОП Піча С. В.

а/с 5026, м. Львів-53, 79053, Україна

e-mail: novsv2016@ukr.net, <https://ns2000.com.ua/>

+38 068-978-94-42, +38 050-337-58-46

Свідоцтво про внесення суб'єкта видавничої справи до Державного реєстру
видавців, виготівників і розповсюджувачів видавничої продукції: серія ДК
№ 5069 від 22.03.2016 року, видане Державним комітетом інформаційної
політики, телебачення та радіомовлення України.