НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

імені Ігоря Сікорського»

Факультет прикладної математики

Кафедра прикладної математики

Звіт

із лабораторної роботи №*2*

із дисципліни «Вступ до баз даних та інформаційних систем»

на тему:

*Даталогічне проектування власної БД. Виконання запитів на мові SQL. Візуалізація запитів*

|  |  |
| --- | --- |
| Виконала: | Керівник: |
| студентка групи КМ-91 | *ст. викладач Бай Ю. П.* |
| *Павловська К.* |  |

Київ — 2021

**ЗМІСТ**

[**Постановка завдання** 2](#_Toc87877792)

[**Теоретичні відомості** 3](#_Toc87877793)

[**Опис виконання завдання** 5](#_Toc87877794)

[**Список літератури** 12](#_Toc87877795)

[ДОДАТОК 1 13](#_Toc87877796)

[ДОДАТОК 2 15](#_Toc87877797)

# **Постановка завдання**

1. Визначитися з тематикою власної БД, обрати датасет на kaggle або data.gov.ua. Перевірити, що обраної тематики ще немає у [Таблиці](https://docs.google.com/spreadsheets/d/1-VSnwiJ7IzvS08lIo8FOZ0fV2uOM6HFs3lL8cc6bFYY/edit?usp=sharing). Записати тематику та посилання на датасет у стовпчики *C, D*.
2. Визначити сутності, атрибути та зв’язки інфологічної моделі БД. *(Кількість сутностей та атрибутів повинна бути мінімально необхідною)*. Записати сутності та їх атрибути у стовпчик *E*.
3. Побудувати ERD реляційної бази даних, використовуючи Draw.io *(нотація для зв’язків між сутностями – “Crow’s Foot”).* Нормалізувати таблиці до 4NF включно. Оформити ERD бази даних у вигляді файлу *erd.drawio.* Назви таблиць БД після нормалізації записати у стовпчик *F*.
4. Написати код для створення таблиць – *create.sql*.
5. Підготувати дані для заповнення таблиць, 5-10 рядків в кожну таблицю. Написати код для заповнення таблиць даними – *populate.sql.*
6. Сформулювати 3 запити (файл *query.sql*), що відповідають заданій структурі:
7. візуалізація – стовпчикова діаграма;
8. візуалізація – кругова діаграма;
9. візуалізація – графік залежності (в разі можна замінити іншою стовпчиковою діаграмою).

Короткий опис запитів записати у стовпчики *G, H, I*.

1. Написати код на Python, що реалізує підключення до PostgreSQL, виконує запити 6a, 6b, 6c та виводить результати їх виконання у консоль. Оформити даний код як файл *main.py*.
2. Реалізувати візуалізацію запитів 6a, 6b, 6c, побудувати dashboard та завантажити його на хостинг plotly. Зберегти файл як *visualization.py.* За бажанням можна виконати дане завдання за допомогою бібліотеки mathplotlib без використання plotly.
3. Створити проект *db\_lab2\_student01* на Github, в якому розташувати файли:

* *erd.drawio*
* *create.sql*
* *populate.sql*
* *query.sql*
* *main.py*
* *visualization.py*

та додати посилання на проект у стовпчик *J*. (***student01*** *=* ***Прізвище студента*** *латиницею*).

# **Теоретичні відомості**

*Нормалізація відношень РБД* – це формальний апарат обмежень на формування таблиць, який дозволяє усунути дублювання даних, забезпечує несуперечність збережених даних та зменшує трудовитрати на ведення бази даних.

Процес нормалізації полягає в розкладі (декомпозиції) початкових відношень бази даних на більш прості відношення. На кожному кроці цього процесу схеми відношень БД приводяться до *нормальних форм*.

*Цілі нормалізації*:

* Забезпечити швидкий доступ до даних
* Видалити непотрібне повторення даних, що може спричинити помилки під час введення, а також призвести до нераціонального використання дискового простору
* Забезпечити цілісність даних, тобто, щоб при зміні одних об'єктів автоматично відбувалася відповідна зміна пов'язаних із ними об'єктів.

Перша нормальна форма (1NF):

1. Кожна таблиця має ключ (простий або композитний) – мінімальний набір стовпчиків, які ідентифікують рядок;
2. Атомарність: кожен атрибут має лише одне значення, а не множину значень.

Друга нормальна форма (2NF):

1. Виконується 1NF;
2. Кожен неключовий атрибут не залежить лише від частини композитного ключа (кожне неключове поле таблиці пов’язане *повною функціональною* залежністю з первинним ключем).

*Функціональною залежністю* між полями А і В називається залежність, при якій кожному значенню А будь-який момент часу відповідає єдине значення В з усіх можливих. *Повною функціональною* залежністю між композитним полем А і полем В називається залежність, при якій поле В залежить функціонально від усього поля A і не залежить функціонально від будь-якої підмножини поля А.

Третя нормальна форма (3NF):

1. Виконується 2NF;
2. Немає *транзитивних залежностей* між неключовими полями.

*Транзитивна функціональ*на залежність між полями А і С існує в тому випадку, якщо поле С функціонально залежить від поля В, а поле B функціонально залежить від поля А, при цьому не існує функціональної залежності поля А від поля В.

Нормальна форма Бойса-Кодда (BCNF):

1. Виконується 3NF;
2. Відсутні функціональні залежності атрибутів первинного ключа від неключових атрибутів

Четверта нормальна форма (4NF):

1. Виконується 3NF/ BCNF;
2. Не містить нетривіальних *багатозначних залежностей*.

*Багатозначна функціональна залежність* між полями визначається таким чином: поле А багатозначно визначає поле В, якщо для кожного значення поля А існує «певна множина» відповідних значень поля В.

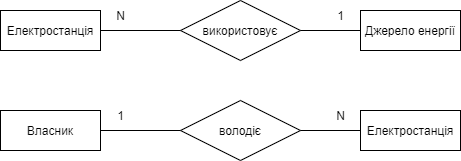
# **Опис виконання завдання**

Обрана тематика БД пов’язана з найбільшими електростанціями у світі. Таблиці міститимуть дані про назви електростанцій, країни, де вони розміщені, географічну широту та довготу, дані про власника електростанції, тип джерела енергії та потужність виробленої електроенергії. Датасет було обрано на kaggle: <https://www.kaggle.com/jaytilala/global-power-plant>.

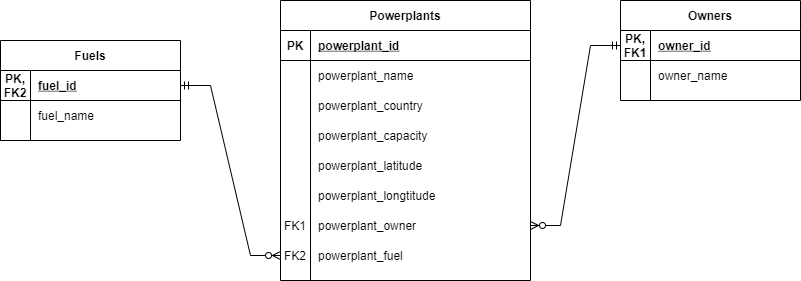
Сутності та атрибути інфологічної моделі БД:

* електростанція (id, назва, країна, широта, довгота, джерело енергії, потужність);
* власник (id, повне ім’я);
* джерело енергії (id, тип джерела енергії).

Зв’язки інфологічної моделі БД:



Після нормалізації таблиць до 4NF включно маємо наступну ERD:



Кожне неключове поле таблиць пов’язане повною функціональною залежністю з первинним ключем, транзитивних залежностей таблиці не містять, багатозначних функціональних залежностей немає.

Код для створення таблиць:

CREATE TABLE Owners(

owner\_id SERIAL PRIMARY KEY,

owner\_name VARCHAR(255)

);

CREATE TABLE Fuels(

fuel\_id SERIAL PRIMARY KEY,

fuel\_name VARCHAR(50)

);

CREATE TABLE Powerplants(

id VARCHAR(12) PRIMARY KEY,

name VARCHAR(255) NOT NULL,

country VARCHAR(255),

capacity NUMERIC,

latitude NUMERIC,

longtitude NUMERIC,

owner INT REFERENCES Owners(owner\_id) ON DELETE CASCADE,

fuel\_type INT REFERENCES Fuels(fuel\_id) ON DELETE RESTRICT

);

Код для наповнення таблиць:

INSERT INTO Fuels(fuel\_name)

VALUES ('oil'), ('hydro'), ('solar'), ('gas'), ('wind'), ('coal'), ('biomass');

INSERT INTO Owners(owner\_name)

VALUES ('Unión Eléctrica'),

('SunRay Power LLC'),

('126 Grove Solar LLC'),

('Skypower Ltd / Sunedison'),

('PAR Renewables'),

('Konoike Pacific'),

('Avidan Energy Solutions'),

('Rochelle Municipal Utilities'),

('158th Fighter Wing'),

('180 Raritan Energy Solutions LLC');

INSERT INTO Powerplants(id, name, country, capacity, latitude, longtitude, owner, fuel\_type)

VALUES ('WRI1002017', '10 De Octubre (nuevitas) Powerplant', 'Cuba', 280.0, 21.5656, -77.2711, 1, 1),

('USA0059371', '12 Applegate Solar LLC', 'United States of America', 1.9, 40.2003, -74.5761, 2, 3),

('USA0060858', '126 Grove Solar LLC', 'United States of America', 2.0, 42.0761, -71.4227, 3, 3),

('CAN0007595', '13th Side Road', 'Canada', 9.5, 42.855, -80.3607, 4, 3),

('GBR0005673', '14 Tullywiggan Road', 'United Kingdom', 1.0, 54.6221, -6.7398, 5, 7),

('USA0057310', '1420 Coil Av #C', 'United States of America', 1.3, 33.7943, -118.2414, 6, 3),

('USA0057458', '145 Talmadge Solar', 'United States of America', 3.8, 40.5358, -74.3913, 7, 3),

('USA0007770', '1515 S Caron Road', 'United States of America', 4.2, 41.9084, -89.0466, 8, 4),

('USA0060542', '158th Fighter Wing Solar Farm', 'United States of America', 1.3, 44.4777, -73.1534, 9, 3),

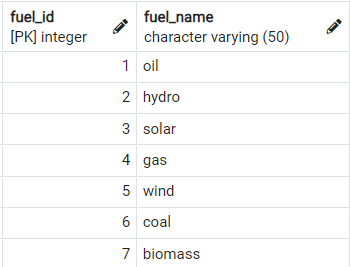
('USA0058187', '180 Raritan Solar', 'United States of America', 1.9, 40.5161, -74.34, 10, 3);

Створені таблиці:

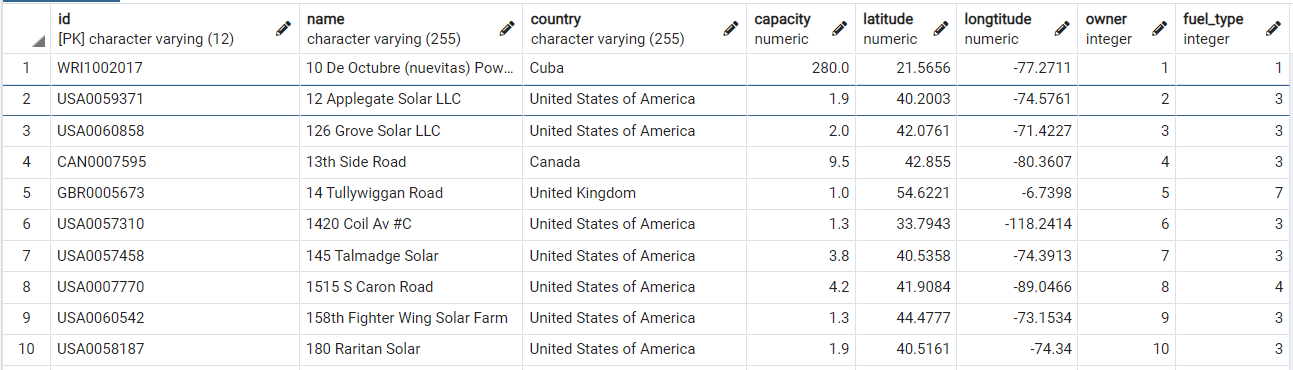
**Owners**



**Fuels**



**Powerplants**



Для стовпчикової діаграми оберемо стовпчики з іменем електростанції та потужністю виробленої електроенергії для країни США і розмістимо дані у порядку спадання потужності. Запит до БД матиме наступний вигляд:

SELECT name, capacity FROM powerplants WHERE country = 'United States of America' ORDER BY capacity DESC;

Кругова діаграма відображатиме кількість електростанцій за типом джерела енергії. Запит до БД наступний:

SELECT fuel\_name, COUNT(powerplants.id) FROM fuels

JOIN powerplants ON fuels.fuel\_id = powerplants.fuel\_type

GROUP BY fuel\_name;

Останній графік відображатиме загальний обсяг потужності виробленої електроенергії для кожного типу джерела енергії, що міститься у таблиці. Запит до БД:

SELECT COUNT(fuel\_name), SUM(powerplants.capacity) FROM fuels

JOIN powerplants ON fuels.fuel\_id = powerplants.fuel\_type

GROUP BY fuel\_name

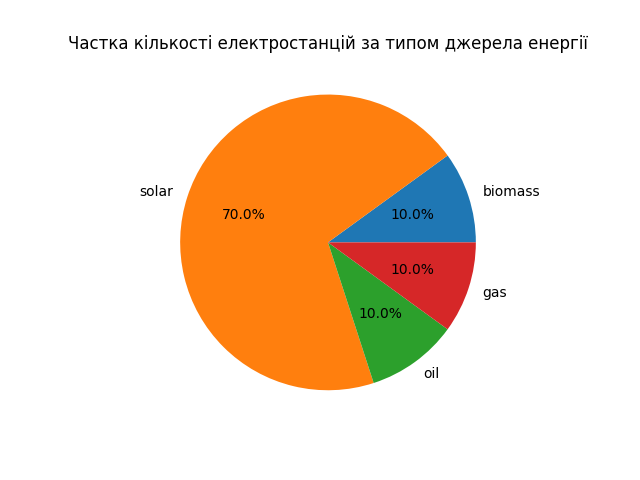
ORDER BY SUM(powerplants.capacity);

Код для візуалізації графіків наведено у додатку 1.

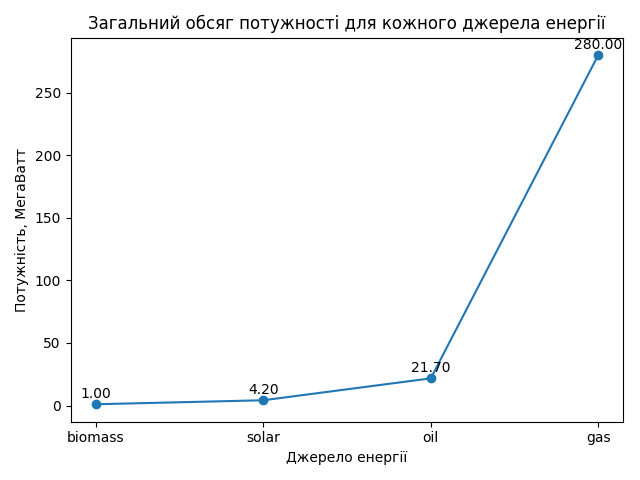
Стовпчикова діаграма:



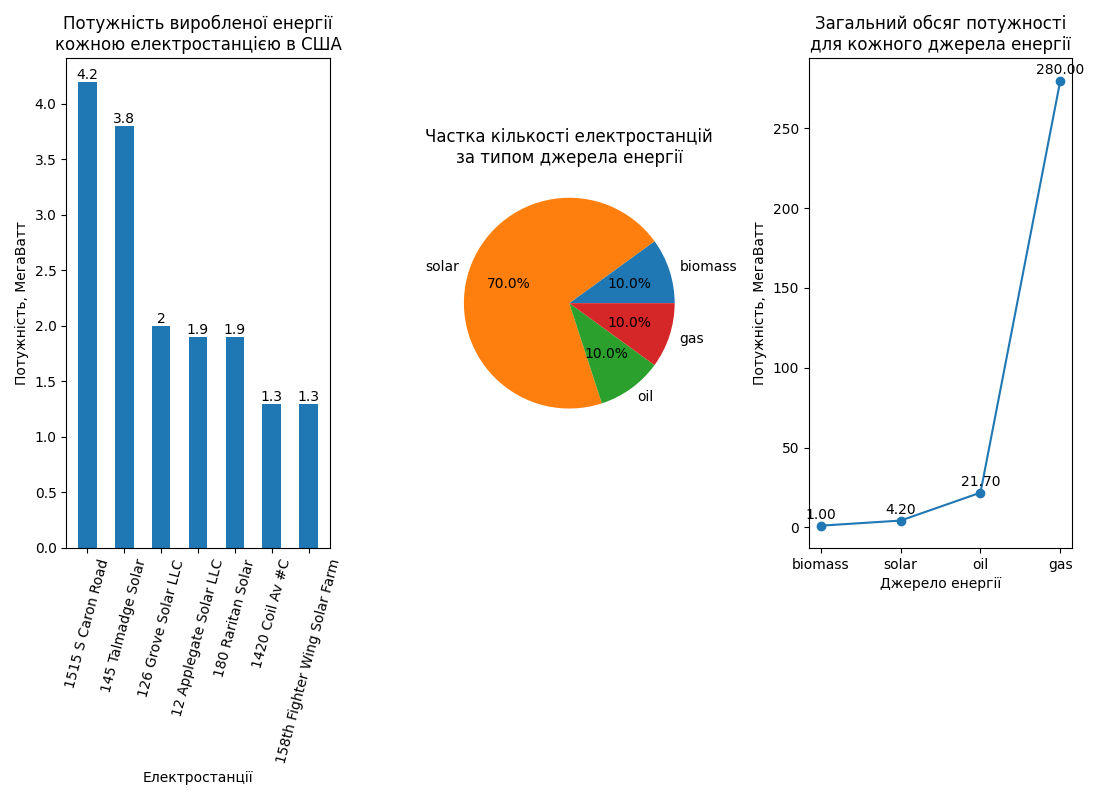
Кругова діаграма:



Графік залежності:



Код для візуалізації дашборду наведено у додатку 2. Дашборд має вигляд:



# **Список літератури**

1. Дейт К. Введение в системы баз данных. – Пер. с англ. – 8-е изд. – К.: Изд. дом «Вильямс», 2006. – 1326 с.

2. Форта Б. Освой самостоятельно SQL. 3-е изд.: Пер. с.англ. – М.: Изд. дом «Вильямс», 2006. – 288 с.

# ДОДАТОК 1

import psycopg2

import matplotlib.pyplot as plt

username = ''

password = ''

database = ''

host = 'localhost'

port = '5432'

query\_1 = '''

SELECT name, capacity FROM powerplants

WHERE country = 'United States of America'

ORDER BY capacity DESC;

'''

query\_2 = '''

SELECT fuel\_name, COUNT(powerplants.id) FROM fuels

JOIN powerplants ON fuels.fuel\_id = powerplants.fuel\_type

GROUP BY fuel\_name;

'''

query\_3 = '''

SELECT COUNT(fuel\_name), SUM(powerplants.capacity) FROM fuels

JOIN powerplants ON fuels.fuel\_id = powerplants.fuel\_type

GROUP BY fuel\_name

ORDER BY SUM(powerplants.capacity);

'''

conn = psycopg2.connect(user=username, password=password, dbname=database, host=host, port=port)

with conn:

cur1 = conn.cursor()

cur1.execute(query\_1)

names = []

capacities = []

for row in cur1:

names.append(row[0])

capacities.append(row[1])

cur2 = conn.cursor()

cur2.execute(query\_2)

fuels = []

fuels\_amount = []

for row in cur2:

fuels.append(row[0])

fuels\_amount.append(row[1])

cur3 = conn.cursor()

cur3.execute(query\_3)

fuel = []

capacity = []

for row in cur3:

fuel.append(row[0])

capacity.append(row[1])

x\_range = range(len(names))

bar = plt.bar(x\_range, capacities, width=0.5)

plt.title('Потужність виробленої енергії кожною електростанцією в США')

plt.xlabel('Електростанції')

plt.xticks(x\_range, names, rotation=75)

plt.ylabel('Потужність, МегаВатт')

plt.bar\_label(bar, label\_type='center')

plt.tight\_layout()

plt.show()

plt.pie(fuels\_amount, labels=fuels, autopct='%1.1f%%')

plt.title('Частка кількості електростанцій за типом джерела енергії')

plt.show()

x\_range = range(len(fuels))

plt.plot(x\_range, capacity, marker='o')

plt.xticks(x\_range, fuels)

plt.title('Загальний обсяг потужності для кожного джерела енергії')

plt.ylabel('Потужність, МегаВатт')

plt.xlabel('Джерело енергії')

for x,y in zip(x\_range,capacity):

label = "{:.2f}".format(y)

plt.annotate(label, (x,y), textcoords="offset points", xytext=(0,4.5), ha='center')

plt.tight\_layout()

plt.show()

# ДОДАТОК 2

import psycopg2

import matplotlib.pyplot as plt

username = ''

password = ''

database = ''

host = 'localhost'

port = '5432'

query\_1 = '''

SELECT name, capacity FROM powerplants

WHERE country = 'United States of America'

ORDER BY capacity DESC;

'''

query\_2 = '''

SELECT fuel\_name, COUNT(powerplants.id) FROM fuels

JOIN powerplants ON fuels.fuel\_id = powerplants.fuel\_type

GROUP BY fuel\_name;

'''

query\_3 = '''

SELECT COUNT(fuel\_name), SUM(powerplants.capacity) FROM fuels

JOIN powerplants ON fuels.fuel\_id = powerplants.fuel\_type

GROUP BY fuel\_name

ORDER BY SUM(powerplants.capacity);

'''

conn = psycopg2.connect(user=username, password=password, dbname=database, host=host, port=port)

with conn:

cur1 = conn.cursor()

cur1.execute(query\_1)

names = []

capacities = []

for row in cur1:

names.append(row[0])

capacities.append(row[1])

figure, (bar\_ax, pie\_ax, graph\_ax) = plt.subplots(1, 3, figsize=(11, 8))

x\_range = range(len(names))

bar = bar\_ax.bar(x\_range, capacities, label='capacities', width=0.5)

bar\_ax.set\_title('Потужність виробленої енергії\nкожною електростанцією в США')

bar\_ax.set\_xticks(x\_range)

bar\_ax.set\_xticklabels(names, fontdict={'rotation': 75})

bar\_ax.set(xlabel='Електростанції', ylabel='Потужність, МегаВатт')

bar\_ax.bar\_label(bar)

cur2 = conn.cursor()

cur2.execute(query\_2)

fuels = []

fuels\_amount = []

for row in cur2:

fuels.append(row[0])

fuels\_amount.append(row[1])

pie\_ax.pie(fuels\_amount, labels=fuels, autopct='%1.1f%%')

pie\_ax.set\_title('Частка кількості електростанцій\nза типом джерела енергії')

cur3 = conn.cursor()

cur3.execute(query\_3)

fuel = []

capacity = []

for row in cur3:

fuel.append(row[0])

capacity.append(row[1])

x\_range = range(len(fuels))

graph\_ax.plot(x\_range, capacity, marker='o')

graph\_ax.set\_xticks(x\_range)

graph\_ax.set\_xticklabels(fuels)

graph\_ax.set\_title('Загальний обсяг потужності\nдля кожного джерела енергії')

graph\_ax.set(xlabel='Джерело енергії', ylabel='Потужність, МегаВатт')

for x, y in zip(x\_range, capacity):

label = "{:.2f}".format(y)

graph\_ax.annotate(label, (x, y), textcoords="offset points", xytext=(0, 4.5), ha='center')

figure.tight\_layout()

plt.show()