МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ «КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Звіт з лабораторної роботи №1

з курсу «Аналіз даних в інформаційних системах»

на тему: «Створення сховища даних»

**Викладач:**

Олійник Ю.О.

**Виконали:**

студенти 2 курсу групи ІП-21 ФІОТ

Голованьов Г.О.

Загребельний О.А.

Київ 2024

**Джерела даних**

Для виконання лабораторної роботи було обрано 3 датасети із сайту <https://www.kaggle.com/>:

1. Загальна інформація про чемпіонат Формули 1 (1950 - 2023): <https://www.kaggle.com/datasets/rohanrao/formula-1-world-championship-1950-2020/data>
2. Детальна інформація про гонщиків: <https://www.kaggle.com/datasets/petalme/f1-drivers-dataset>
3. Інформація про інциденти під час перегонів: <https://www.kaggle.com/datasets/jtrotman/formula-1-race-events>

Спільними вимірами у цих джерелах є гонщики, етапи чемпіонтату, команди, країни, роки.

**Модель stage зони та основного сховища**

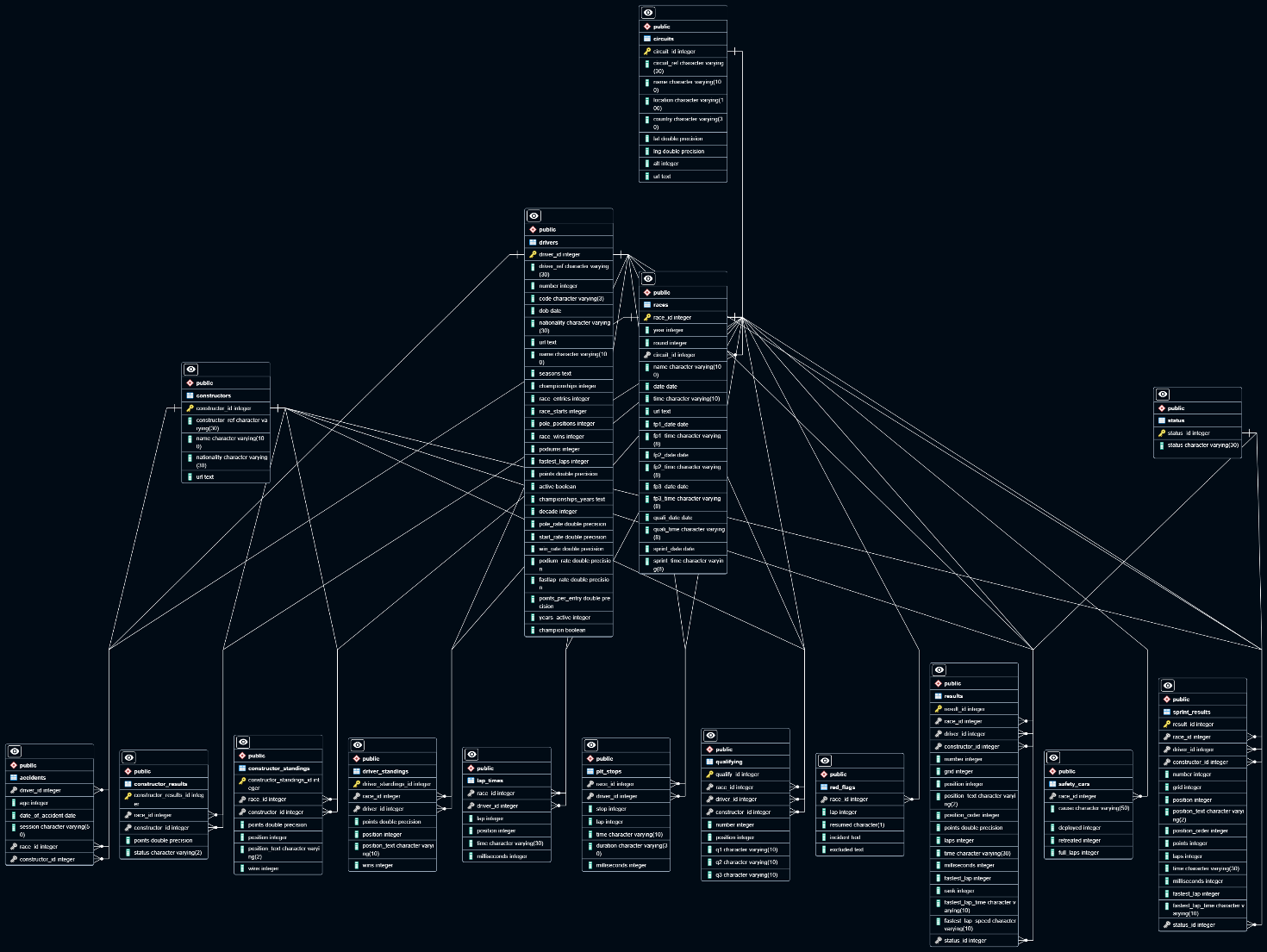
****

Рис. 1 – Cховище за типом «сніжинка»

Скрипти створення основного сховища знаходяться у додатку А.

Опис таблиць сховища:

1. Accidents – таблиця призначена для зберігання інформації про аварії з фатальними наслідками для гонщика.
2. Circuits - таблиця призначена для зберігання інформації про гоночні треки.
3. Constructor\_results - таблиця призначена для зберігання інформації про результати команди у тій чи іншій гонці.
4. Constructor\_standings - таблиця призначена для зберігання інформації про результати кубку конструкторів після тої чи іншої гонки.
5. Constructors - таблиця призначена для зберігання інформації про команди.
6. Driver\_standings - таблиця призначена для зберігання інформації про чемпіонат пілотів після тої чи іншої гонки.
7. Drivers - таблиця призначена для зберігання інформації про гонщиків.
8. Lap\_times - таблиця призначена для зберігання інформації про проїхані кола гонщиків.
9. Pit\_stops - таблиця призначена для зберігання інформації про піт-стопи під час перегонів.
10. Qualifying - таблиця призначена для зберігання інформації про результати кваліфікаційних сесій.
11. Races – таблиця призначена для зберігання інформації про результати гонок.
12. Red\_flags - таблиця призначена для зберігання інформації про червоні прапори під час сесій.
13. Results - таблиця призначена для зберігання інформації про результати гонщиків в конкретній гонці.
14. Safety\_cars - таблиця призначена для зберігання інформації про режим автомобіля безпеки під час сесій.
15. Sprint\_results - таблиця призначена для зберігання інформації про гоночні спринти.
16. Status - таблиця призначена для зберігання інформації про статус, з яким гонщик закінчив перегони.

**ETL засоби**

Файли csv з обраних джерел було імпортовано як вхідні дані. Щоб поєднати дані з різних датасетів були використані скрипти на Python, що наведені в додатку Б, а після отримання коректних вихідних файлів було дані з них було імпортовано у сховище засобами СУБД PostgreSQL.

**Додаток А**

(Скрипти для створення сховища)

*/\*---------------------------------------------------*

*CIRCUITS*

*---------------------------------------------------\*/*

CREATE TABLE circuits (

    circuit\_id SERIAL PRIMARY KEY,

    circuit\_ref VARCHAR(30),

    name VARCHAR(100),

    location VARCHAR(100),

    country VARCHAR(30),

    lat FLOAT,

    lng FLOAT,

    alt INT,

    url TEXT

);

*/\*---------------------------------------------------*

*CONSTRUCTORS*

*---------------------------------------------------\*/*

CREATE TABLE constructors (

    constructor\_id SERIAL PRIMARY KEY,

    constructor\_ref VARCHAR(30),

    name VARCHAR(100),

    nationality VARCHAR(30),

    url TEXT

);

*/\*---------------------------------------------------*

*DRIVERS*

*---------------------------------------------------\*/*

CREATE TABLE drivers (

    driver\_id SERIAL PRIMARY KEY,

    driver\_ref VARCHAR(30),

    number INT,

    code VARCHAR(3),

    dob DATE,

    nationality VARCHAR(30),

    url TEXT,

    name VARCHAR(100),

    seasons TEXT,

    championships INT,

    race\_entries INT,

    race\_starts INT,

    pole\_positions INT,

    race\_wins INT,

    podiums INT,

    fastest\_laps INT,

    points FLOAT,

    active BOOLEAN,

    championships\_years TEXT,

    decade INT,

    pole\_rate FLOAT,

    start\_rate FLOAT,

    win\_rate FLOAT,

    podium\_rate FLOAT,

    fastlap\_rate FLOAT,

    points\_per\_entry FLOAT,

    years\_active INT,

    champion BOOLEAN

);

*/\*---------------------------------------------------*

*STATUS*

*---------------------------------------------------\*/*

CREATE TABLE status (

    status\_id SERIAL PRIMARY KEY,

    status VARCHAR(30)

);

*/\*---------------------------------------------------*

*RACES*

*---------------------------------------------------\*/*

CREATE TABLE races (

    race\_id SERIAL PRIMARY KEY,

    year INT,

    round INT,

    circuit\_id INT REFERENCES circuits(circuit\_id),

    name VARCHAR(100),

    date DATE,

    time VARCHAR(10),

    url TEXT,

    fp1\_date date,

    fp1\_time VARCHAR(8),

    fp2\_date date,

    fp2\_time VARCHAR(8),

    fp3\_date date,

    fp3\_time VARCHAR(8),

    quali\_date date,

    quali\_time VARCHAR(8),

    sprint\_date date,

    sprint\_time VARCHAR(8)

);

*/\*---------------------------------------------------*

*LAP TIMES*

*---------------------------------------------------\*/*

CREATE TABLE lap\_times (

    race\_id INT REFERENCES races(race\_id),

    driver\_id INT REFERENCES drivers(driver\_id),

    lap INT,

    position INT,

    time VARCHAR(30),

    milliseconds INT

);

*/\*---------------------------------------------------*

*DRIVER STANDINGS*

*---------------------------------------------------\*/*

CREATE TABLE driver\_standings (

    driver\_standings\_id SERIAL PRIMARY KEY,

    race\_id INT REFERENCES races(race\_id),

    driver\_id INT REFERENCES drivers(driver\_id),

    points FLOAT,

    position INT,

    position\_text VARCHAR(10),

    wins INT

);

*/\*---------------------------------------------------*

*CONSTRUCTOR STANDINGS*

*---------------------------------------------------\*/*

CREATE TABLE constructor\_standings (

    constructor\_standings\_id SERIAL PRIMARY KEY,

    race\_id INT REFERENCES races(race\_id),

    constructor\_id INT REFERENCES constructors(constructor\_id),

    points FLOAT,

    position INT,

    position\_text VARCHAR(2),

    wins INT

);

*/\*---------------------------------------------------*

*CONSTRUCTOR RESULTS*

*---------------------------------------------------\*/*

CREATE TABLE constructor\_results (

    constructor\_results\_id SERIAL PRIMARY KEY,

    race\_id INT REFERENCES races(race\_id),

    constructor\_id INT REFERENCES constructors(constructor\_id),

    points FLOAT,

    status VARCHAR(2)

);

*/\*---------------------------------------------------*

*PIT STOPS*

*---------------------------------------------------\*/*

CREATE TABLE pit\_stops (

    race\_id INT REFERENCES races(race\_id),

    driver\_id INT REFERENCES drivers(driver\_id),

    stop INT,

    lap INT,

    time VARCHAR(10),

    duration VARCHAR(30),

    milliseconds INT

);

*/\*---------------------------------------------------*

*QUALIFYING*

*---------------------------------------------------\*/*

CREATE TABLE qualifying (

    qualify\_id SERIAL PRIMARY KEY,

    race\_id INT REFERENCES races(race\_id),

    driver\_id INT REFERENCES drivers(driver\_id),

    constructor\_id INT REFERENCES constructors(constructor\_id),

    number INT,

    position INT,

    q1 VARCHAR(10),

    q2 VARCHAR(10),

    q3 VARCHAR(10)

);

*/\*---------------------------------------------------*

*RESULTS*

*---------------------------------------------------\*/*

CREATE TABLE results (

    result\_id SERIAL PRIMARY KEY,

    race\_id INT REFERENCES races(race\_id),

    driver\_id INT REFERENCES drivers(driver\_id),

    constructor\_id INT REFERENCES constructors(constructor\_id),

    number INT,

    grid INT,

    position INT,

    position\_text VARCHAR(2),

    position\_order INT,

    points INT,

    laps INT,

    time VARCHAR(30),

    milliseconds INT,

    fastest\_lap INT,

    rank INT,

    fastest\_lap\_time VARCHAR(10),

    fastest\_lap\_speed VARCHAR(10),

    status\_id INT REFERENCES status(status\_id)

);

*/\*---------------------------------------------------*

*SPRINT RESULTS*

*---------------------------------------------------\*/*

CREATE TABLE sprint\_results (

    result\_id SERIAL PRIMARY KEY,

    race\_id INT REFERENCES races(race\_id),

    driver\_id INT REFERENCES drivers(driver\_id),

    constructor\_id INT REFERENCES constructors(constructor\_id),

    number INT,

    grid INT,

    position INT,

    position\_text VARCHAR(2),

    position\_order INT,

    points INT,

    laps INT,

    time VARCHAR(30),

    milliseconds INT,

    fastest\_lap INT,

    fastest\_lap\_time VARCHAR(10),

    status\_id INT REFERENCES status(status\_id)

);

*/\*---------------------------------------------------*

*ACCIDENTS*

*---------------------------------------------------\*/*

CREATE TABLE accidents (

    driver\_id INT REFERENCES drivers(driver\_id),

    age INT,

    date\_of\_accident DATE,

    session VARCHAR(50),

    race\_id INT REFERENCES races(race\_id),

    constructor\_id INT REFERENCES constructors(constructor\_id)

);

*/\*---------------------------------------------------*

*SAFETY CARS*

*---------------------------------------------------\*/*

CREATE TABLE safety\_cars (

    race\_id INT REFERENCES races(race\_id),

    cause VARCHAR(50),

    deployed INT,

    retreated INT,

    full\_laps INT

);

*/\*---------------------------------------------------*

*RED FLAGS*

*---------------------------------------------------\*/*

CREATE TABLE red\_flags (

    race\_id INT REFERENCES races(race\_id),

    lap INT,

    resumed CHAR(1),

    incident TEXT,

    excluded TEXT

);

**Додаток Б**

(Скрипти Python)

accidents.py

*import* pandas *as* pd

*import* numpy *as* np

*from* datetime *import* datetime, timedelta

accidents\_df = pd.read\_csv('input\_csv/fatal\_accidents\_drivers.csv')

drivers\_df = pd.read\_csv('input\_csv/drivers.csv')

races\_df = pd.read\_csv('output\_csv/races.csv')

constructors\_df = pd.read\_csv('output\_csv/constructors.csv')

def concat\_name(*row*):

    name\_parts = *row*['Driver'].split()

    surname = name\_parts[-1]

    forename = ' '.join(name\_parts[:-1])

*return* pd.Series([forename, surname])

def concat\_race(*row*):

    race = str(*row*['year']) + ' ' + *row*['name']

*return* pd.Series([race])

accidents\_df[['forename', 'surname']] = accidents\_df.apply(concat\_name, *axis*=1)

races\_df['Event'] = races\_df.apply(concat\_race, *axis*=1)

constructors\_df.rename(*columns*={'name': 'Car'}, *inplace*=True)

merged\_df = pd.merge(drivers\_df[['driverId', 'forename', 'surname']], accidents\_df, *on*=['forename', 'surname'], *how*='inner')

merged\_df = pd.merge(merged\_df, races\_df[['raceId', 'Event']], *on*=['Event'], *how*='inner')

merged\_df = pd.merge(merged\_df, constructors\_df[['constructorId', 'Car']], *on*=['Car'], *how*='inner')

merged\_df.drop(['forename', 'surname', 'Driver', 'Car', 'Event'], *axis*=1, *inplace*=True)

merged\_df['Date Of Accident'] = pd.to\_datetime(merged\_df['Date Of Accident'], *format*='mixed')

future\_dates = merged\_df['Date Of Accident'] > datetime.now()

merged\_df.loc[future\_dates, 'Date Of Accident'] -= timedelta(*days*=36525)

merged\_df['Age'].replace('', np.nan, *inplace*=True)

merged\_df['Age'] = merged\_df['Age'].astype(float).astype('Int64')

merged\_df.to\_csv('output\_csv/accidents.csv', *index*=False)

drivers\_spliting.py

*import* pandas *as* pd

*import* csv

df\_split\_names = pd.read\_csv('input\_csv/drivers.csv')

df\_not\_split\_names = pd.read\_csv('input\_csv/F1Drivers\_Dataset.csv')

def concat\_name(*row*):

    name = *row*['forename'] + ' ' + *row*['surname']

*return* pd.Series([name])

df\_split\_names[['Driver']] = df\_split\_names.apply(concat\_name, *axis*=1)

merged\_df = pd.merge(df\_split\_names, df\_not\_split\_names, *on*=['Driver'], *how*='inner')

merged\_df = merged\_df.drop(*columns*=['forename', 'surname', 'Nationality'])

merged\_df.to\_csv('output\_csv/drivers.csv', *index*=False, *quoting*=csv.QUOTE\_NONNUMERIC)

red\_flags.py

*import* pandas *as* pd

red\_flags\_df = pd.read\_csv('input\_csv/red\_flags.csv')

races\_df = pd.read\_csv('output\_csv/races.csv')

def concat\_race(*row*):

    race = str(*row*['year']) + ' ' + *row*['name']

*return* pd.Series([race])

races\_df['Race'] = races\_df.apply(concat\_race, *axis*=1)

merged\_df = pd.merge(races\_df[['raceId', 'Race']], red\_flags\_df, *on*=['Race'], *how*='inner')

merged\_df.drop(['Race'], *axis*=1, *inplace*=True)

merged\_df.to\_csv('output\_csv/red\_flags.csv', *index*=False)

safety\_cars.py

*import* pandas *as* pd

*import* numpy *as* np

safety\_cars\_df = pd.read\_csv('input\_csv/safety\_cars.csv')

races\_df = pd.read\_csv('output\_csv/races.csv')

def concat\_race(*row*):

    race = str(*row*['year']) + ' ' + *row*['name']

*return* pd.Series([race])

races\_df['Race'] = races\_df.apply(concat\_race, *axis*=1)

merged\_df = pd.merge(races\_df[['raceId', 'Race']], safety\_cars\_df, *on*=['Race'], *how*='inner')

merged\_df.drop(['Race'], *axis*=1, *inplace*=True)

merged\_df['Retreated'].replace('', np.nan, *inplace*=True)

merged\_df['Retreated'] = merged\_df['Retreated'].astype(float).astype('Int64')

merged\_df.to\_csv('output\_csv/safety\_cars.csv', *index*=False)