МIНIСТЕРСТВО ВИЩОЇ ОСВIТИ УКРАЇНИ   
НАЦIОНАЛЬНИЙ ТЕХНІЧНИЙ УНIВЕРСИТЕТ УКРАЇНИ   
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»

Кафедра автоматизації енергетичних процесів

Лабораторна робота №2

“Вивчення основ мови SQL”

Курс: ПРОЄКТУВАННЯ ТА РОЗРОБКА БАЗ ДАНИХ

Виконала:

Студентка групи ТА-13, НН ІАТЕ

Аксьонова Я.В.

Перевірили:

Любицький С.В.

Київ 2023

1. **Завдання:**

Створити базу даних згідно моделі отриманій в попередні роботі та вивчити основні оператори мови SQL.

1. **Текст SQL створення всіх таблиць бази даних. (CREATE TABLE, ALTER TABLE) з відповідними обмеженнями.:**

CREATE TABLE Cinema (

CinemaName varchar(50) NOT NULL,

CinemaAdress varchar(100) UNIQUE NOT NULL,

HallQuantity int check(HallQuantity>0),

CinemaWorkerQuantity int NOT NULL,

CinemaID int PRIMARY KEY

);

CREATE TABLE CinemaWorker (

FullName varchar(50) NOT NULL,

Position varchar(40) NOT NULL,

Salary int NOT NULL,

Workschedule varchar(3) DEFAULT '5/2',

CinemaWorkerId int PRIMARY KEY,

CONSTRAINT FK\_CinemaWorkerId FOREIGN KEY (CinemaWorkerId) REFERENCES Cinema(CinemaID)

);

CREATE TABLE Hall (

HallName varchar(40) UNIQUE NOT NULL,

SeatsQuantity int check(SeatsQuantity>0),

HallID int PRIMARY KEY,

CONSTRAINT FK\_HallID FOREIGN KEY (HallID) REFERENCES Cinema(CinemaID)

);

CREATE TABLE Film (

FilmName varchar(40) UNIQUE NOT NULL,

Duration int NOT NULL,

Producer varchar(30) NOT NULL,

Studio varchar(30) NOT NULL,

SeanceQuantity int check(SeanceQuantity>0),

Genre varchar(50),

FilmID int PRIMARY KEY

);

CREATE TABLE Seanse (

Film int NOT NULL,

Hall int NOT NULL,

SeanseTime smalldatetime DEFAULT CURRENT\_TIMESTAMP,

SeanseID int PRIMARY KEY,

CONSTRAINT FK\_FilmSeanse FOREIGN KEY (Film) REFERENCES Film(FilmID),

CONSTRAINT FK\_HallSeanse FOREIGN KEY (Hall) REFERENCES Hall(HallID)

);

CREATE TABLE Customer(

FullName varchar(40) NOT NULL,

BirthDate date NOT NULL,

CustomerID int PRIMARY KEY

);

CREATE TABLE Ticket (

SeatNumber int NOT NULL,

Seanse int NOT NULL,

Customer int NOT NULL,

Price int check(Price>50),

TicketId int PRIMARY KEY,

CONSTRAINT FK\_TicketSeanse FOREIGN KEY (Seanse) REFERENCES Seanse(SeanseID),

CONSTRAINT FK\_TicketCustomer FOREIGN KEY (Customer) REFERENCES Customer(CustomerID)

);

CREATE TABLE Purchase (

Ticket int PRIMARY KEY,

Customer int NOT NULL,

PurchaseDate datetime DEFAULT CURRENT\_TIMESTAMP,

CONSTRAINT FK\_TicketPurchase FOREIGN KEY (Ticket) REFERENCES Ticket(TicketID),

CONSTRAINT FK\_CustomerPurchase FOREIGN KEY (Customer) REFERENCES Customer(CustomerID)

);

CREATE TABLE Studio (

StudioID varchar(40) PRIMARY KEY,

StudioCountry varchar(20)

);

CREATE TABLE Sponsore (

Film int NOT NULL,

Studio varchar(40) NOT NULL,

CONSTRAINT FK\_SponsoreStudio FOREIGN KEY (Studio) REFERENCES Studio(StudioID),

CONSTRAINT FK\_SponsoreFilm FOREIGN KEY (Film) REFERENCES Film(FilmID),

);

CREATE TABLE FilmWorker (

Positon varchar(20) NOT NULL,

FullName varchar(40) NOT NULL,

WorkerID int PRIMARY KEY

);

CREATE TABLE Activity (

Film int NOT NULL,

FilmWorker int NOT NULL,

CONSTRAINT FK\_ActivityFilmWorker FOREIGN KEY (FilmWorker) REFERENCES FilmWorker(WorkerID),

CONSTRAINT FK\_ActivityFilm FOREIGN KEY (Film) REFERENCES Film(FilmID),

);



Рис.1 Результат команди CREATE TABLE

SELECT \* From FilmWorker

ALTER TABLE FilmWorker ADD CONSTRAINT Salaryy check (WorkerID>0) ;

ALTER TABLE FilmWorker DROP CONSTRAINT Salaryy;

ALTER TABLE FilmWorker DROP COLUMN Positon;

ALTER TABLE FilmWorker ADD Position char(20);

Зображення, що містить текст, знімок екрана, Шрифт, число

Автоматично згенерований опис

Рис.2 Результат команди ALTER TABLE

1. **Приклади виконання операторів INSERT, UPDATE, DELETE:**

INSERT INTO Cinema VALUES('Ukraine2', 'Downstreet 867',5, 150,0)

INSERT INTO Cinema VALUES('Multiplex', 'Gnata Chotkevycha, 1В',3, 100,1)

INSERT INTO Cinema VALUES('Kioto', 'Milutenka, 19', 1, 50,2)

INSERT INTO Cinema VALUES('Wizoria', 'Architektora Verbyckoho', 2, 80,3)

INSERT INTO Cinema VALUES('KINO42', 'Konstantynivska, 11B', 3, 120,4)

INSERT INTO Cinema VALUES('Butterfly Nyvky', 'Danyla Serbakivskoho, 58', 1, 20,5)

Зображення, що містить текст, знімок екрана, Шрифт, число

Автоматично згенерований опис

Рис.3 Результат команди INSERT

DELETE FROM Cinema

WHERE HallQuantity > 4

Зображення, що містить текст, знімок екрана, Шрифт, число

Автоматично згенерований опис

Рис.4 Результат команди DELETE

UPDATE Cinema

SET CinemaName = 'Kioto'

WHERE CinemaID = 2;

Зображення, що містить текст, знімок екрана, Шрифт, число

Автоматично згенерований опис

Рис.5 Результат команди UPDATE

1. **Приклади виконання оператора SELECT з різними прикладами фільтрації - порівняння, LIKE, BETWEEN:**

SELECT \* FROM Cinema

WHERE CinemaID >=2;

Зображення, що містить текст, знімок екрана, Шрифт, ряд

Автоматично згенерований опис

Рис.6 Результат команди SELECT з прикладом фільтрації порівняння

SELECT \* FROM Cinema

WHERE CinemaName LIKE '%Kioto%';



Рис.7 Результат команди SELECT з прикладом фільтрації LIKE

SELECT \* FROM Cinema

WHERE CinemaID BETWEEN 2 AND 5;Зображення, що містить текст, знімок екрана, Шрифт, ряд

Автоматично згенерований опис

Рис.8 Результат команди SELECT з прикладом фільтрації BETWEEN

1. **Приклади операції проєкції та її різновидів - (DISTINCT, TOP, AS ), сортування (ORDER BY, OFFSET, FETCH):**

SELECT DISTINCT Genre FROM Film;

Зображення, що містить текст, знімок екрана, Шрифт, число

Автоматично згенерований опис

Рис.9 Результат команди проекції DISTINCT

SELECT TOP 3 \* FROM Film ORDER BY Duration;

Зображення, що містить текст, знімок екрана, Шрифт, ряд

Автоматично згенерований опис

Рис.10 Результат команди проекції TOP з сортуванням ORDER BY

SELECT FilmName AS 'FilmName', Duration AS 'Duration' FROM Film;

Зображення, що містить текст, знімок екрана, Шрифт, ряд

Автоматично згенерований опис

Рис.11 Результат команди проекції AS

SELECT \* FROM Film ORDER BY FilmID DESC;

Зображення, що містить текст, знімок екрана, Шрифт, ряд

Автоматично згенерований опис

Рис.12 Результат команди проекції DISTINCT з сортуванням DESC

SELECT \* FROM Film ORDER BY FilmID OFFSET 2 ROWS FETCH NEXT 3 ROWS ONLY;

Зображення, що містить текст, знімок екрана, ряд, Шрифт

Автоматично згенерований опис

Рис.13 Результат сортування OFFSET та FETCH

1. **Приклади операторів UNION, EXCEPT, INTERSECT**:

SELECT FilmID AS ID, Duration AS Час FROM Film

UNION

SELECT SeanseID, SeanseTime FROM Seanse;

Зображення, що містить текст, знімок екрана, Шрифт, число

Автоматично згенерований опис

Рис.14 Результат оператора UNION

SELECT FilmID FROM Film

EXCEPT

SELECT Film FROM Seanse;

Зображення, що містить знімок екрана, текст, Шрифт, білий

Автоматично згенерований опис

Рис.15 Результат оператора EXCEPT

SELECT FilmID FROM Film

INTERSECT

SELECT Film FROM Seanse;

Зображення, що містить знімок екрана, Шрифт, ряд, число

Автоматично згенерований опис

Рис.16 Результат оператора INTERSECT

1. **Операція умовного з’єднання. Види та типи. (INNER, LEFT, RIGHT, FULL, CROSS JOIN)**:

SELECT \*

FROM Film

INNER JOIN Seanse

ON Film.FilmID = Seanse.Film;

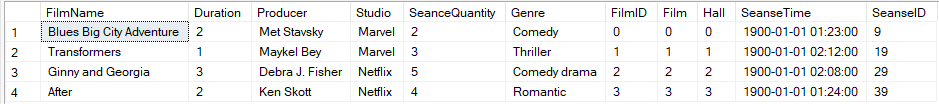


Рис.17 Результат операції INNER JOIN

SELECT \*

FROM Film

LEFT JOIN Seanse

ON Film.FilmID = Seanse.Film;

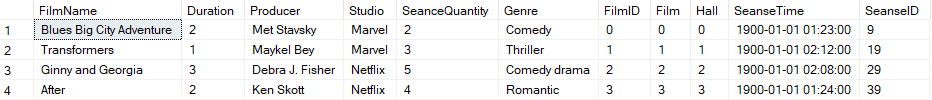


Рис.18 Результат операції LEFT JOIN

SELECT \*

FROM Cinema

FULL JOIN Hall

ON Cinema.CinemaID = Hall.HallID;

Зображення, що містить текст, знімок екрана, Шрифт, ряд

Автоматично згенерований опис

Рис.19 Результат операції FULL JOIN

SELECT \*

FROM Cinema

CROSS JOIN Hall;

Зображення, що містить текст, Шрифт, число, ряд

Автоматично згенерований опис

Рис.20 Результат операції CROSS JOIN

1. **Вкладені запити. EXIST, IN, ANY, ALL**:

Для наступних операцій використаємо цю таблицю та таблицю Cinema з першого завдання

SELECT \*

FROM Cinema c

WHERE EXISTS (

SELECT \*

FROM FilmWorker cw

WHERE cw.WorkerID = c.CinemaID

AND cw.Positon = 'Manager'

);

Зображення, що містить текст, знімок екрана, Шрифт, ряд

Автоматично згенерований опис

Рис.21 Результат вкладеного запиту EXISTS

SELECT \*

FROM Film

WHERE Genre IN (

SELECT DISTINCT Genre

FROM Film

WHERE Duration > 1

);

Зображення, що містить текст, знімок екрана, Шрифт, ряд

Автоматично згенерований опис

Рис.22 Результат вкладеного запиту IN

SELECT \*

FROM Film

WHERE Duration > ANY (

SELECT Duration

FROM Film

WHERE Genre = 'Comedy'

);



Рис.23 Результат вкладеного запиту ANY

SELECT \*

FROM Film

WHERE Duration > ALL (

SELECT Duration

FROM Film

WHERE Genre = 'Comedy'

);



Рис.24 Результат вкладеного запиту ALL

1. **Групування та агрегатні функції (GROUP BY, HAVING, AVG, MIN, MAX, SUM, COUNT)**:

SELECT Genre, COUNT(\*) as SeanceQuantity

FROM Film

GROUP BY Genre;

Зображення, що містить текст, знімок екрана, Шрифт, ряд

Автоматично згенерований опис

Рис.25 Результат групування GROUP BY

SELECT AVG(SeatsQuantity) AS AVGSeatsQuantity FROM Hall;



Рис.26 Результат функції AVG

SELECT HallName, SUM(SeatsQuantity) as TotalSeatsQuantity

FROM Hall

GROUP BY HallName

HAVING SUM(SeatsQuantity) > 50;

Зображення, що містить текст, знімок екрана, Шрифт, число

Автоматично згенерований опис

Рис.27 Результат групування HAVING та функції SUM

SELECT MAX(SeatsQuantity) as MaxSeatsQuantity

FROM Hall;



Рис.28 Результат функції MAX

SELECT COUNT(\*) as TotalCount

FROM Hall;



Рис.29 Результат функції COUNT

1. **Висновки**:

Під час виконання лабораторної роботи ми розглянули основні команди SQL для роботи з реляційними базами даних. Ми почали з створення таблиць, додавання та видалення стовпців, а також обмежень для таблиць. Потім ми перейшли до виконання операцій (CREATE TABLE, ALTER TABLE) за допомогою операторів SELECT, INSERT, UPDATE та DELETE.

Ми також вивчили команди для фільтрації даних (LIKE, BETWEEN), сортування (ORDER BY, OFFSET, FETCH), проекції (DISTINCT, TOP, AS), операцій об'єднання (UNION, EXCEPT, INTERSECT) та умовного з’єднання (INNER, LEFT, RIGHT, FULL, CROSS JOIN). Також ми розглянули вкладені запити з використанням операторів EXIST, IN, ANY та ALL.

Заключною частиною лабораторної роботи стали групування та агрегатні функції (GROUP BY, HAVING, AVG, MIN, MAX, SUM, COUNT), які дозволяють отримувати згруповані та підраховані значення з бази даних.

Отже, ми розглянули основні команди мови SQL, необхідні для роботи з реляційними базами даних.

1. **Індивідуальне завдання**:

Створити запит, який знайде продюсера якого подивилася найбільша кількість глядачів.

Зображення, що містить текст, знімок екрана, монітор, програмне забезпечення

Автоматично згенерований опис