МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНОМУ УНІВЕРСИТЕТІ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”

Кафедра систем штучного інтелекту

Лабораторна робота №2

з дисципліни

«Організація баз даних та знань»

Виконав:

студент групи КН-208

Шегда Микола

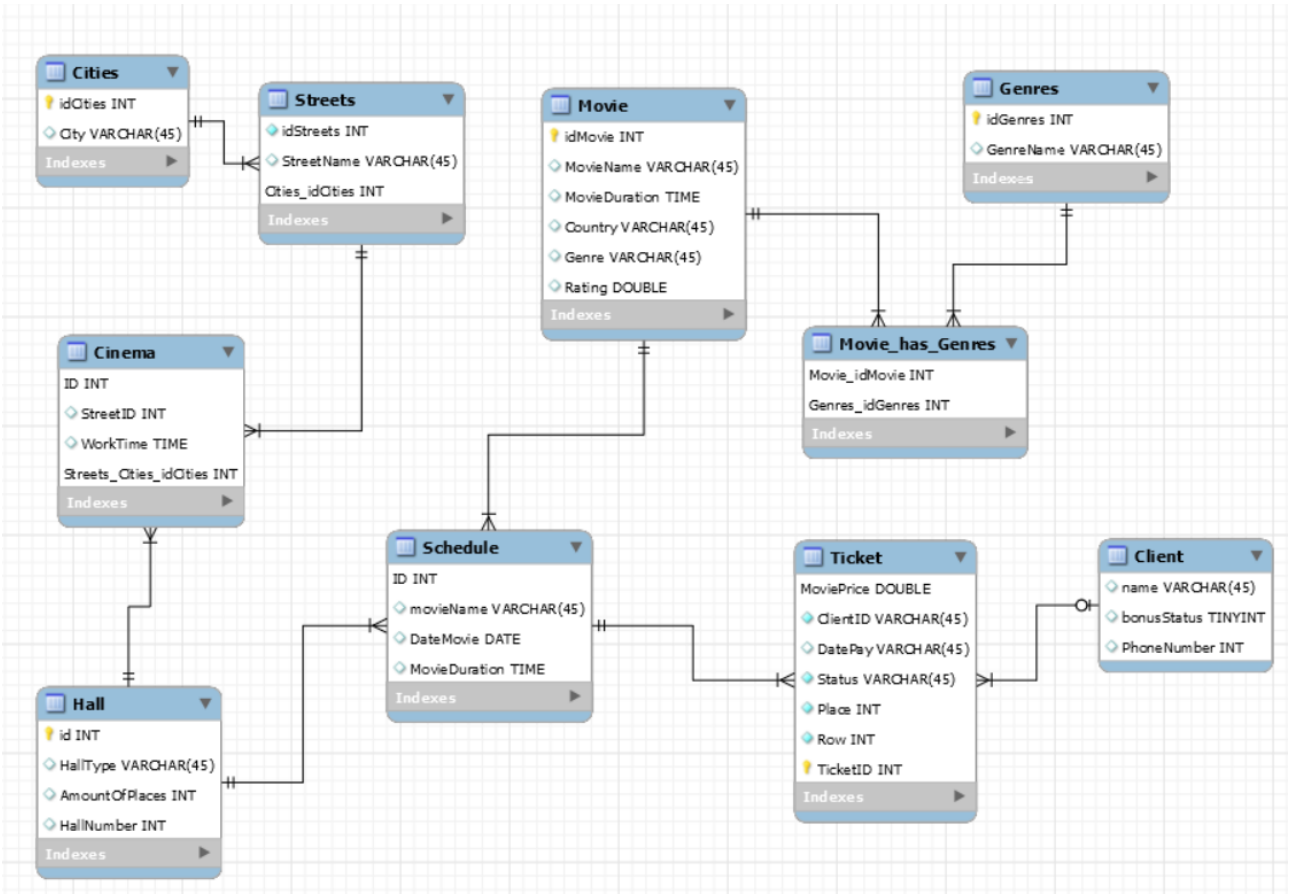
Викладач:

Мельникова Н. І.

Львів – 2019 р.

Мета роботи: Побудувати даталогічну модель бази даних; визначити типи, розмірності та обмеження полів; визначити обмеження таблиць; розробити SQL запити для
створення спроектованих таблиць.

На основі діаграми побудувати Базу Даних.



Створюємо базу даних

```
CREATE DATABASE Test;
```

Далі створюємо таблицю Movie ,

присвоюємо атрибуту ‘idMovie’ Первинний ключ, автоінкрементацію та not null

```
CREATE TABLE Movie(
    idMovie INT NOT NULL IDENTITY(1,1),
    MovieName VARCHAR(99),
    MovieDuration TIME,
    Country VARCHAR(99),
    Genre VARCHAR(99),
    Rating INT,
  PRIMARY KEY (idMovie)
);
```

Після того створюємо таблицю Hall,

присвоюємо атрибуту ‘id’ Первинний ключ, автоінкрементацію та not null

```
CREATE TABLE Hall(
    id INT NOT NULL IDENTITY(1,1),
    Halltype VARCHAR(10),
    Places INT,
    HallNumber INT,
    PRIMARY KEY (id)
);
```

Наступною таблицею створюємо Schedule,

присвоюємо атрибуту ‘id’ Первинний ключ, автоінкрементацію та not null

Foreign Key ‘id’ з’єднуємо з ‘idMovie` з таблиці Movie

‘id’ з’єднуємо з ‘id’ з таблиці Hall

```
CREATE TABLE Schedule(
    id INT NOT NULL IDENTITY(1,1),
    MovieName VARCHAR(99),
    DateMovie DATE,
    MovieDuration TIME,
    PRIMARY KEY (id),
    CONSTRAINT FK_Schedule_Movie FOREIGN KEY (id)
    REFERENCES Movie (idMovie) ON DELETE NO ACTION ON UPDATE NO ACTION,
    CONSTRAINT FK_Schedule_Hall FOREIGN KEY (id)
    REFERENCES Hall (id) ON DELETE NO ACTION ON UPDATE NO ACTION
);
```

Створюємо таблицю Client

Присвоюємо атрибуту ‘id’ Первинний ключ, автоінкрементацію та not null

```
CREATE TABLE Client(
    id INT NOT NULL IDENTITY(1,1),
    ClientName VARCHAR(99),
    Bonus TINYINT,
    PhoneNumbre INT
    PRIMARY KEY(id)
);
```

Далі створюємо таблицю Ticket

Присвоюємо атрибуту ‘id’ Первинний ключ, автоінкрементацію та not null

Атрибуту ‘MoviePrice’ присвоюємо Foreign Key та з’єднуємо з ‘id’ з таблиці Schedule та Client.

```
CREATE TABLE Ticket(
    MoviePrice INT NOT NULL,
    ClientID VARCHAR(99) NOT NULL,
    DatePay VARCHAR(99) NULL,
    TicketStatus VARCHAR(99) NOT NULL,
    Place INT NOT NULL,
    RowPlace INT NOT NULL,
    TicketID INT NOT NULL IDENTITY(1,1),
    PRIMARY KEY (TicketID, MoviePrice),
    CONSTRAINT FK_Ticket_Schedule FOREIGN KEY (MoviePrice)
    REFERENCES Schedule (id) ON DELETE NO ACTION ON UPDATE NO ACTION,
    CONSTRAINT FK_Ticket_Client FOREIGN KEY (MoviePrice)
    REFERENCES Client (id) ON DELETE NO ACTION ON UPDATE NO ACTION
);
```

Наступною таблицею буде Cities,

Присвоюємо атрибуту ‘id’ Первинний ключ, автоінкрементацію та not null

```
CREATE TABLE Cities(
    id INT NOT NULL IDENTITY(1,1),
    City VARCHAR(99),
    PRIMARY KEY (id)
);
```

Далі створюємо таблицю з назвою Streets

Присвоюємо атрибуту ‘id’ Первинний ключ, автоінкрементацію та not null

Атрибут ‘CitiesId’ робимо Foreign Key та з’єднуємо з ‘id’ з таблиці Cities

```
CREATE TABLE Streets(
    id INT NOT NULL IDENTITY(1,1),
    StreetName VARCHAR(99),
    CitiesId INT NOT NULL,
    PRIMARY KEY (CitiesId),
    CONSTRAINT FK_Streets_Cities FOREIGN KEY (CitiesId)
    REFERENCES Cities (id) ON DELETE NO ACTION ON UPDATE NO ACTION
);
```

Робимо таблицю Cinema

Присвоюємо атрибуту ‘id’ Первинний ключ, автоінкрементацію та not null

Тут а нас два Foreign Key:

‘id’ з’єднуємо з ‘id’ з Hall

А ‘StreetsId’ зєднуємо з ‘CitiesId’ з таблиці Streets

```
CREATE TABLE Cinema(
    id INT NOT NULL IDENTITY(1,1),
    WorkTime TIME,
    StreetsId INT,
    CONSTRAINT FK_Cinema_Hall FOREIGN KEY (id)
    REFERENCES Hall (id) ON DELETE NO ACTION ON UPDATE NO ACTION,
    CONSTRAINT FK_Cinema_Streets FOREIGN KEY (StreetsId)
    REFERENCES Streets (CitiesId) ON DELETE NO ACTION ON UPDATE NO ACTION
);
```

Створюємо таблицю Genres

Присвоюємо атрибуту ‘id’ Первинний ключ, автоінкрементацію та not null

```
CREATE TABLE Genres(
    id INT NOT NULL IDENTITY(1,1),
    GenreName VARCHAR(99),
    PRIMARY KEY (id)
);
```

Далі створюємо таблицю Movie_has_Genres для зв’язку ManyToMany між таблицями Movie та Genres

Присвоюємо MovieId Foreign Key та з’єднуємо з ‘idMovie’ в таблиці Movie,

Також присвоюємо Foreign Key GenreId та з’єднуємо з ‘id’ в таблиці Genres

```
CREATE TABLE Movie_has_Genres(
    MovieId INT NOT NUll,
    GenreId INT NOT NULL,
    PRIMARY KEY (MovieId, GenreId),
    CONSTRAINT FK_Movie_has_Genres_Movie FOREIGN KEY (MovieId)
    REFERENCES Movie (idMovie) ON DELETE NO ACTION ON UPDATE NO ACTION,
    CONSTRAINT FK_Movie_has_Genres_Genres FOREIGN KEY (GenreId)
    REFERENCES Genres (id) ON DELETE NO ACTION ON UPDATE NO ACTION
);
```

Код запиту:

```
IF EXISTS(SELECT * FROM sys.databases where name = 'Test')
DROP DATABASE Test;

CREATE DATABASE Test;

use Test;

CREATE TABLE Movie(
       idMovie INT NOT NULL IDENTITY(1,1),
       MovieName VARCHAR(99),
       MovieDuration TIME,
       Country VARCHAR(99),
       Genre VARCHAR(99),
       Rating INT,
  PRIMARY KEY (idMovie)
);

CREATE TABLE Hall(
```

```
	id INT NOT NULL IDENTITY(1,1),
	Halltype VARCHAR(10),
	Places INT,
	HallNumber INT,
	PRIMARY KEY (id)
);

CREATE TABLE Schedule(
	id INT NOT NULL IDENTITY(1,1),
	MovieName VARCHAR(99),
	DateMovie DATE,
	MovieDuration TIME,
	PRIMARY KEY (id),
	CONSTRAINT FK_Schedule_Movie FOREIGN KEY (id)
    REFERENCES Movie (idMovie) ON DELETE NO ACTION ON UPDATE NO ACTION,
    CONSTRAINT FK_Schedule_Hall FOREIGN KEY (id)
    REFERENCES Hall (id) ON DELETE NO ACTION ON UPDATE NO ACTION
);

CREATE TABLE Client(
	id INT NOT NULL IDENTITY(1,1),
	ClientName VARCHAR(99),
	Bonus TINYINT,
	PhoneNumbre INT
	PRIMARY KEY(id)
);

CREATE TABLE Ticket(
	MoviePrice INT NOT NULL,
    ClientID VARCHAR(99) NOT NULL,
    DatePay VARCHAR(99) NULL,
    TicketStatus VARCHAR(99) NOT NULL,
    Place INT NOT NULL,
    RowPlace INT NOT NULL,
    TicketID INT NOT NULL IDENTITY(1,1),
    PRIMARY KEY (TicketID, MoviePrice),
    CONSTRAINT FK_Ticket_Schedule FOREIGN KEY (MoviePrice)
    REFERENCES Schedule (id) ON DELETE NO ACTION ON UPDATE NO ACTION,
    CONSTRAINT FK_Ticket_Client FOREIGN KEY (MoviePrice)
    REFERENCES Client (id) ON DELETE NO ACTION ON UPDATE NO ACTION
);

CREATE TABLE Cities(
	id INT NOT NULL IDENTITY(1,1),
	City VARCHAR(99),
	PRIMARY KEY (id)
);

CREATE TABLE Streets(
	id INT NOT NULL IDENTITY(1,1),
	StreetName VARCHAR(99),
	CitiesId INT NOT NULL,
	PRIMARY KEY (CitiesId),
	CONSTRAINT FK_Streets_Cities FOREIGN KEY (CitiesId)
	REFERENCES Cities (id) ON DELETE NO ACTION ON UPDATE NO ACTION
);

CREATE TABLE Cinema(
	id INT NOT NULL IDENTITY(1,1),
	WorkTime TIME,
	StreetsId INT,
	CONSTRAINT FK_Cinema_Hall FOREIGN KEY (id)
	REFERENCES Hall (id) ON DELETE NO ACTION ON UPDATE NO ACTION,
	CONSTRAINT FK_Cinema_Streets FOREIGN KEY (StreetsId)
	REFERENCES Streets (CitiesId) ON DELETE NO ACTION ON UPDATE NO ACTION
);

CREATE TABLE Genres(
```

```
	id INT NOT NULL IDENTITY(1,1),
	GenreName VARCHAR(99),
	PRIMARY KEY (id)
);

CREATE TABLE Movie_has_Genres(
	MovieId INT NOT NUll,
	GenreId INT NOT NULL,
	PRIMARY KEY (MovieId, GenreId),
	CONSTRAINT FK_Movie_has_Genres_Movie FOREIGN KEY (MovieId)
	REFERENCES Movie (idMovie) ON DELETE NO ACTION ON UPDATE NO ACTION,
	CONSTRAINT FK_Movie_has_Genres_Genres FOREIGN KEY (GenreId)
	REFERENCES Genres (id) ON DELETE NO ACTION ON UPDATE NO ACTION
);
```

Висновок

На цій лабораторній роботі я завершив моделювання і засобами SQL створив базу даних, що складається з семи таблиць.