

**Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки**

Лабораторна робота №5

з дисципліни
«Бази даних»

Виконала:

студентка групи ІМ-42
Шалак Анастасія Володимирівна
номер у списку групи: 30

Перевірив:

Русінов В. В.

Київ 2025

Лабораторна робота 5: Нормалізація бази даних

1. Початкова схема (після Лаб 1–3)

Hall, Movie, Genres, Movies_Genres,

Showings(showing_id, show_time TIMESTAMP, price),

Schedule, Clients, Tickets(show_date TIMESTAMP, ...)

2. Виявлені проблеми та аномалії

Таблиця	Проблема	Аномалія оновлення
Showings	show_time → price	Зміна ціни на 18:00 вимагає оновлення всіх рядків
Tickets	show_date дублює частину show_time	Можлива неузгодженість дати сеансу

3. Функціональні залежності (ФЗ)

Showings (до нормалізації):

showing_id → show_time, price

show_time → price ← транзитивна залежність

Tickets:

ticket_id → show_date, schedule_id, client_id, seat

schedule_id → showing_id → show_time → show_date ← транзитивна

4. Перевірка нормальних форм (початкова схема)

Таблиця	1NF	2NF	3NF	Причина
Hall	Yes	Yes	Yes	—
Movie	Yes	Yes	Yes	—
Genres	Yes	Yes	Yes	—

Movies_Genres	Yes	Yes	Yes	—
Clients	Yes	Yes	Yes	—
Schedule	Yes	Yes	Yes	—
Showings	Yes	Yes	No	show_time → price
Tickets	Yes	Yes	No	show_date залежить від schedule_id

Висновок: схема не в повній 3NF.

5. Нормалізація до 3NF (покроково)

Крок 1 → 1NF (вже виконано)

Крок 2 → 2NF (вже виконано)

Крок 3 → 3NF

Виділяємо ціну за часом сеансу в окрему таблицю

```
CREATE TABLE Showtime_Prices (
    show_time TIME PRIMARY KEY,
    price NUMERIC(8,2) NOT NULL
);
```

Оновлюємо Showings

```
CREATE TABLE Showings_New (
    showing_id SERIAL PRIMARY KEY,
    show_date DATE NOT NULL,
    show_time TIME NOT NULL REFERENCES Showtime_Prices(show_time)
);
```

Прибираємо show_date з Tickets

```
ALTER TABLE Tickets DROP COLUMN IF EXISTS show_date;
```

6. Фінальна схема (повна 3NF)

-- Ціни за часом сеансу

```
CREATE TABLE Showtime_Prices (  
    show_time TIME PRIMARY KEY,  
    price    NUMERIC(8,2) NOT NULL  
);
```

-- Сеанси

```
CREATE TABLE Showings (  
    showing_id SERIAL PRIMARY KEY,  
    show_date  DATE NOT NULL,  
    show_time  TIME NOT NULL REFERENCES Showtime_Prices(show_time)  
);
```

-- Розклад

```
CREATE TABLE Schedule (  
    schedule_id SERIAL PRIMARY KEY,  
    hall_id     INT NOT NULL REFERENCES Hall(hall_id) ON DELETE CASCADE,  
    showing_id  INT NOT NULL REFERENCES Showings(showing_id) ON DELETE  
CASCADE,  
    movie_id    INT NOT NULL REFERENCES Movie(movie_id) ON DELETE  
CASCADE  
);
```

-- Квитки

CREATE TABLE Tickets (

ticket_id SERIAL PRIMARY KEY,

schedule_id INT NOT NULL REFERENCES Schedule(schedule_id) ON DELETE CASCADE,

client_id INT NOT NULL REFERENCES Clients(client_id) ON DELETE CASCADE,

seat VARCHAR(10) NOT NULL

);

7. Переваги нової схеми

Аномалія	Було	Стало
Зміна ціни на 18:00	Оновлення багатьох рядків	1 рядок у Showtime_Prices
Додавання нової ціни	Дублювання в Showings	1 INSERT в Showtime_Prices
Неузгодженість дати сеансу	Можлива в Tickets	Неможлива — дата тільки в Showings
Зміна часу сеансу	Ризик невідповідності в Tickets	Автоматично через FK

8. Повний скрипт міграції

-- 1. Таблиця цін за часом

CREATE TABLE Showtime_Prices (

show_time TIME PRIMARY KEY,

price NUMERIC(8,2) NOT NULL

);

-- 2. Заповнення цін

```
INSERT INTO Showtime_Prices (show_time, price)
SELECT DISTINCT show_time::TIME, price FROM Showings
ON CONFLICT DO NOTHING;
```

-- 3. Нова таблиця Showings

```
CREATE TABLE Showings_New (
    showing_id SERIAL PRIMARY KEY,
    show_date DATE NOT NULL,
    show_time TIME NOT NULL REFERENCES Showtime_Prices(show_time)
);
```

-- 4. Перенос даних

```
INSERT INTO Showings_New (showing_id, show_date, show_time)
SELECT showing_id, show_time::DATE, show_time::TIME FROM Showings;
```

-- 5. Заміна таблиці

```
DROP TABLE Showings;
ALTER TABLE Showings_New RENAME TO Showings;
```

-- 6. Оновлення FK в Schedule

```
ALTER TABLE Schedule
```

```
DROP CONSTRAINT IF EXISTS schedule_showing_id_fkey;
```

```
ALTER TABLE Schedule
```

```
ADD CONSTRAINT schedule_showing_id_fkey
```

```
FOREIGN KEY (showing_id) REFERENCES Showings(showing_id) ON DELETE  
CASCADE;
```

```
-- 7. Прибирання show_date з Tickets
```

```
ALTER TABLE Tickets DROP COLUMN IF EXISTS show_date;
```

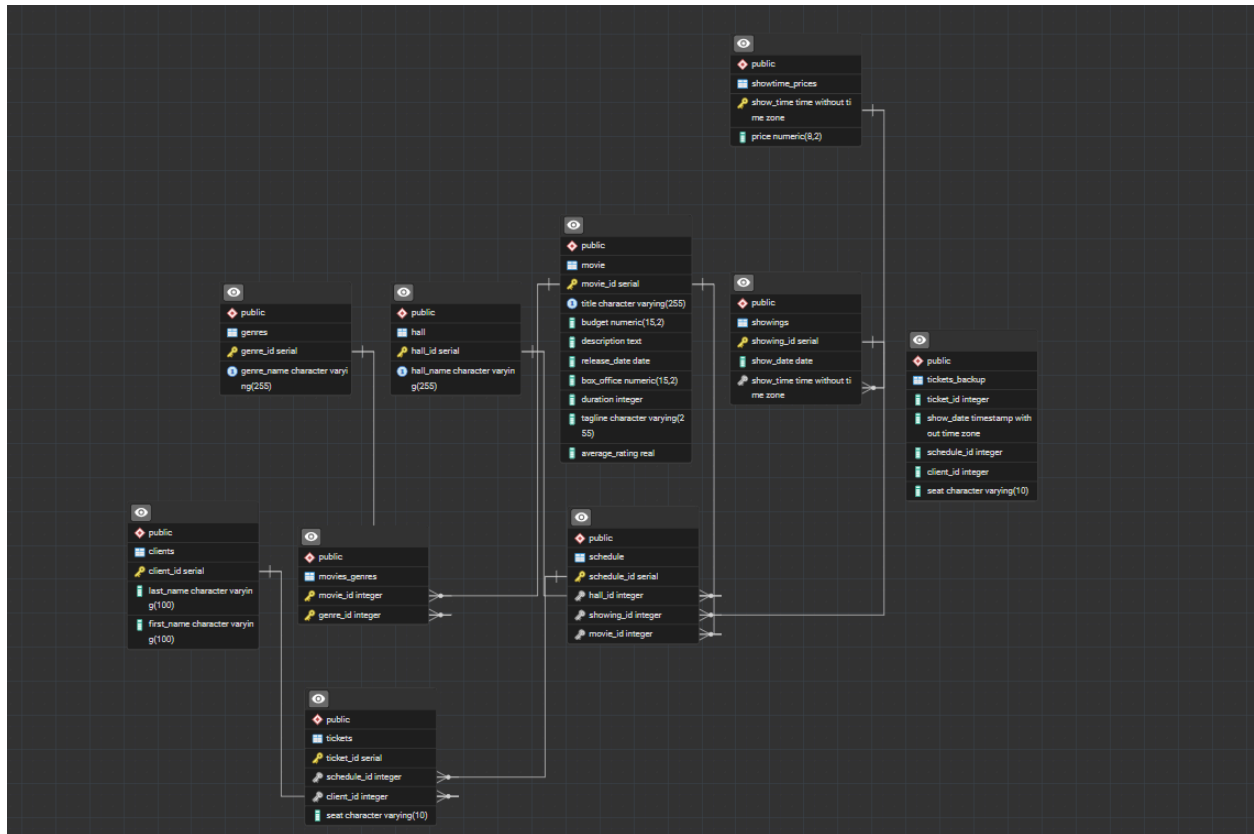
```
-- 8. Перевірка
```

```
SELECT 'Showtime_Prices' AS tbl, COUNT(*) FROM Showtime_Prices UNION ALL
```

```
SELECT 'Showings', COUNT(*) FROM Showings UNION ALL
```

```
SELECT 'Tickets', COUNT(*) FROM Tickets;
```

9. Оновлена ER-діаграма



10. Висновок

- Початкова схема: порушувала 3NF через транзитивні залежності
- Після нормалізації: повна 3NF
- Усунено надлишковість і аномалії оновлення
- Підвищено цілісність даних
- Схема готова до масштабування (легко додавати знижки, нічні тарифи тощо)