Dokumentacja Bazy Danych Kina

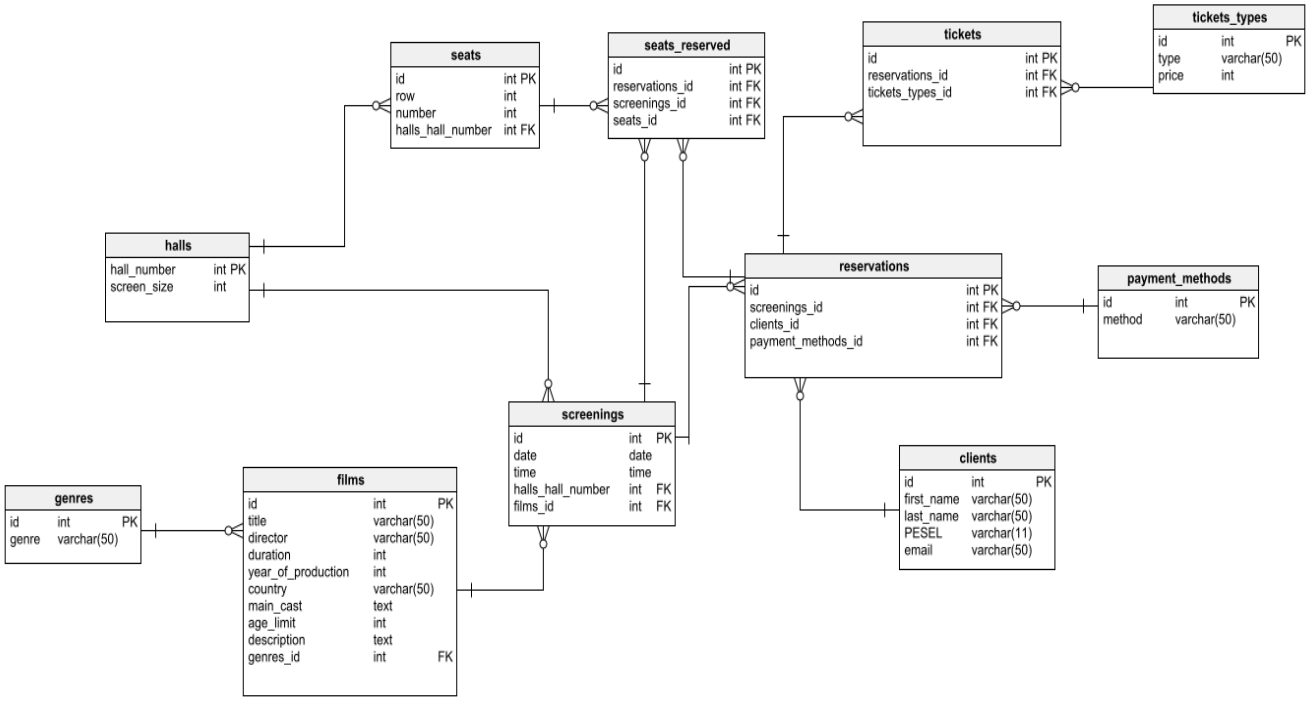
Projekt na zajęcia Relacyjne Bazy Danych

Mikołaj Kawczyński
s25441
gr. 28c

1. Temat projektu

Relacyjna Baza Danych Kina mająca na celu wspomaganie działania tego rodzaju placówki. Ma za zadanie dostarczać szereg wspierających pracę w kinie funkcji, takich jak system rezerwacji seansów, czy też przechowywanie informacji o filmach, seansach i klientach. Projekt stworzony został wykorzystując strukturalny język zapytań SQL oraz oprogramowanie DataGrip.

2. Diagram (wizualna prezentacja bazy danych)



Vertabelo

3. Zapytanie tworzące strukturę bazy danych

```sql
CREATE TABLE clients (
    id int NOT NULL,
    first_name varchar(50) NOT NULL,
    last_name varchar(50) NOT NULL,
    PESEL varchar(11) NOT NULL,
    email varchar(50) NOT NULL,
    CONSTRAINT clients_pk PRIMARY KEY (id)
);

-- Table: films
CREATE TABLE films (
    id int NOT NULL,
    title varchar(50) NOT NULL,
    director varchar(50) NOT NULL,
    duration int NOT NULL,
    year_of_production int NOT NULL,
    country varchar(50) NOT NULL,
    main_cast text NOT NULL,
    age_limit int NOT NULL,
    description text NOT NULL,
    genres_id int NOT NULL,
    CONSTRAINT films_pk PRIMARY KEY (id)
);

-- Table: genres
CREATE TABLE genres (
    id int NOT NULL,
    genre varchar(50) NOT NULL,
    CONSTRAINT genres_pk PRIMARY KEY (id)
);

-- Table: halls
CREATE TABLE halls (
    hall_number int NOT NULL,
    screen_size int NOT NULL,
    CONSTRAINT halls_pk PRIMARY KEY (hall_number)
);

-- Table: payment_methods
CREATE TABLE payment_methods (
    id int NOT NULL,
    method varchar(50) NOT NULL,
    CONSTRAINT payment_methods_pk PRIMARY KEY (id)
);

-- Table: reservations
CREATE TABLE reservations (
    id int NOT NULL,
    screenings_id int NOT NULL,
    clients_id int NOT NULL,
    payment_methods_id int NOT NULL,
    CONSTRAINT reservations_pk PRIMARY KEY (id)
);

-- Table: screenings
CREATE TABLE screenings (
```

```
    id int NOT NULL,
    date date NOT NULL,
    time time NOT NULL,
    halls_hall_number int NOT NULL,
    films_id int NOT NULL,
    CONSTRAINT screenings_pk PRIMARY KEY (id)
);

-- Table: seats
CREATE TABLE seats (
    id int NOT NULL,
    row int NOT NULL,
    number int NOT NULL,
    halls_hall_number int NOT NULL,
    CONSTRAINT seats_pk PRIMARY KEY (id)
);

-- Table: seats_reserved
CREATE TABLE seats_reserved (
    id int NOT NULL,
    reservations_id int NOT NULL,
    screenings_id int NOT NULL,
    seats_id int NOT NULL,
    CONSTRAINT seats_reserved_pk PRIMARY KEY (id)
);

-- Table: tickets
CREATE TABLE tickets (
    id int NOT NULL,
    reservations_id int NOT NULL,
    tickets_types_id int NOT NULL,
    CONSTRAINT tickets_pk PRIMARY KEY (id)
);

-- Table: tickets_types
CREATE TABLE tickets_types (
    id int NOT NULL,
    type varchar(50) NOT NULL,
    price int NOT NULL,
    CONSTRAINT tickets_types_pk PRIMARY KEY (id)
);

-- foreign keys
-- Reference: films_genres (table: films)
ALTER TABLE films ADD CONSTRAINT films_genres FOREIGN KEY films_genres
(genres_id)
    REFERENCES genres (id);

-- Reference: reservations_clients (table: reservations)
ALTER TABLE reservations ADD CONSTRAINT reservations_clients FOREIGN KEY
reservations_clients (clients_id)
    REFERENCES clients (id);

-- Reference: reservations_payment_methods (table: reservations)
ALTER TABLE reservations ADD CONSTRAINT reservations_payment_methods
FOREIGN KEY reservations_payment_methods (payment_methods_id)
    REFERENCES payment_methods (id);

-- Reference: reservations_screenings (table: reservations)
ALTER TABLE reservations ADD CONSTRAINT reservations_screenings FOREIGN KEY
reservations_screenings (screenings_id)
```

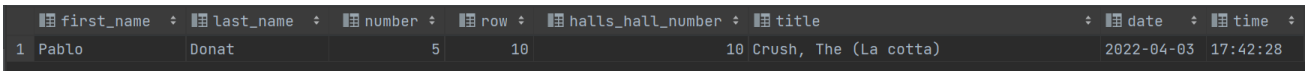
```
    REFERENCES screenings (id);

-- Reference: screenings_films (table: screenings)
ALTER TABLE screenings ADD CONSTRAINT screenings_films FOREIGN KEY
screenings_films (films_id)
    REFERENCES films (id);

-- Reference: screenings_halls (table: screenings)
ALTER TABLE screenings ADD CONSTRAINT screenings_halls FOREIGN KEY
screenings_halls (halls_hall_number)
    REFERENCES halls (hall_number);

-- Reference: seats_halls (table: seats)
ALTER TABLE seats ADD CONSTRAINT seats_halls FOREIGN KEY seats_halls
(halls_hall_number)
    REFERENCES halls (hall_number);

-- Reference: seats_reseved_reservations (table: seats_reserved)
ALTER TABLE seats_reserved ADD CONSTRAINT seats_reseved_reservations
FOREIGN KEY seats_reseved_reservations (reservations_id)
    REFERENCES reservations (id);

-- Reference: seats_reseved_screenings (table: seats_reserved)
ALTER TABLE seats_reserved ADD CONSTRAINT seats_reseved_screenings FOREIGN
KEY seats_reseved_screenings (screenings_id)
    REFERENCES screenings (id);

-- Reference: seats_reseved_seats (table: seats_reserved)
ALTER TABLE seats_reserved ADD CONSTRAINT seats_reseved_seats FOREIGN KEY
seats_reseved_seats (seats_id)
    REFERENCES seats (id);

-- Reference: tickets_reservations (table: tickets)
ALTER TABLE tickets ADD CONSTRAINT tickets_reservations FOREIGN KEY
tickets_reservations (reservations_id)
    REFERENCES reservations (id);

-- Reference: tickets_tickets_types (table: tickets)
ALTER TABLE tickets ADD CONSTRAINT tickets_tickets_types FOREIGN KEY
tickets_tickets_types (tickets_types_id)
    REFERENCES tickets_types (id);

-- End of file.
```

4. Najczęściej wykorzystywane zapytania

1) Zapytanie zwracające najpotrzebniejsze informacje o seansie, na który dany klient się wybiera (w tym wypadku klient o id = 1). Zwraca imię i nazwisko klienta, a także numer sali, w której będzie odbywał się seans, rząd i numer miejsca, datę i godzinę seansu oraz tytuł filmu:

```
SELECT clients.first_name, clients.last_name, seats.number,
seats.row, seats.halls_hall_number, films.title,
screenings.date, screenings.time FROM clients
JOIN reservations on clients.id = reservations.clients_id
JOIN seats_reserved on reservations.id =
seats_reserved.reservations_id
JOIN seats on seats.id = seats_reserved.seats_id
JOIN screenings on screenings.id = reservations.screenings_id
JOIN films on films.id = screenings.films_id
WHERE clients.id = 1;
```

Wynik zapytania:

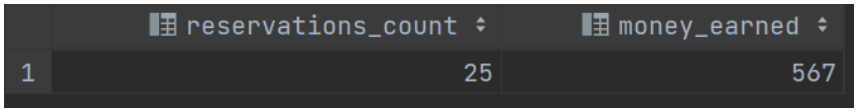


	first_name	last_name	number	row	halls_hall_number	title	date	time
1	Pablo	Donat	5	10	10	Crush, The (La cotta)	2022-04-03	17:42:28

2) Zapytanie zwracające ilość rezerwacji dokonanych w danym okresie czasu oraz ilość zarobionych na biletach pieniędzy:

```
SELECT COUNT(reservations.id) AS reservations_count,
SUM(tt.price) AS money_earned FROM reservations
JOIN screenings s on s.id = reservations.screenings_id
JOIN tickets t on reservations.id = t.reservations_id
JOIN tickets_types tt on tt.id = t.tickets_types_id
WHERE s.date >= '2022-06-01' && s.date < '2022-07-01';
```

Wynik zapytania:



	reservations_count	money_earned
1	25	567

3) Zapytanie zwracające wszystkie seanse, które odbywają się danego dnia:

```
SELECT title, time, hall_number FROM films
JOIN screenings s ON films.id = s.films_id
JOIN halls h ON h.hall_number = s.halls_hall_number
WHERE date='2022-05-04';
```

Wynik zapytania:

	title	time	hall_number
1	Roll Bounce	21:00:48	3
2	Manhattan Project, The	06:26:33	5
3	Enduring Love	06:32:55	5
4	Loves of Carmen, The	16:05:22	5
5	Phantasm	11:10:10	7
6	Trailer Park Boys: Countdown to Liquor Day	11:07:38	7
7	Thrilla in Manila	19:20:30	8
8	Rock 'n' Roll Nightmare	00:35:01	8
9	Errors of the Human Body	23:16:37	10

4) Zapytanie zwracające wszystkie najważniejsze informacje o danym filmie:

```
SELECT films.title, films.director, films.duration,
films.year_of_production, films.country, films.main_cast,
films.age_limit, films.description, genre FROM films
JOIN genres g on g.id = films.genres_id
WHERE films.id = 1;
```

Wynik zapytania:

	title	director	duration	year_of_production	country	main_cast	age_limit	descrip...	genre
1	U.S. Seals	Dulci Mushett	593	2006	Peru	Mannie Bland	5	nullam sit ame…	Sci-Fi