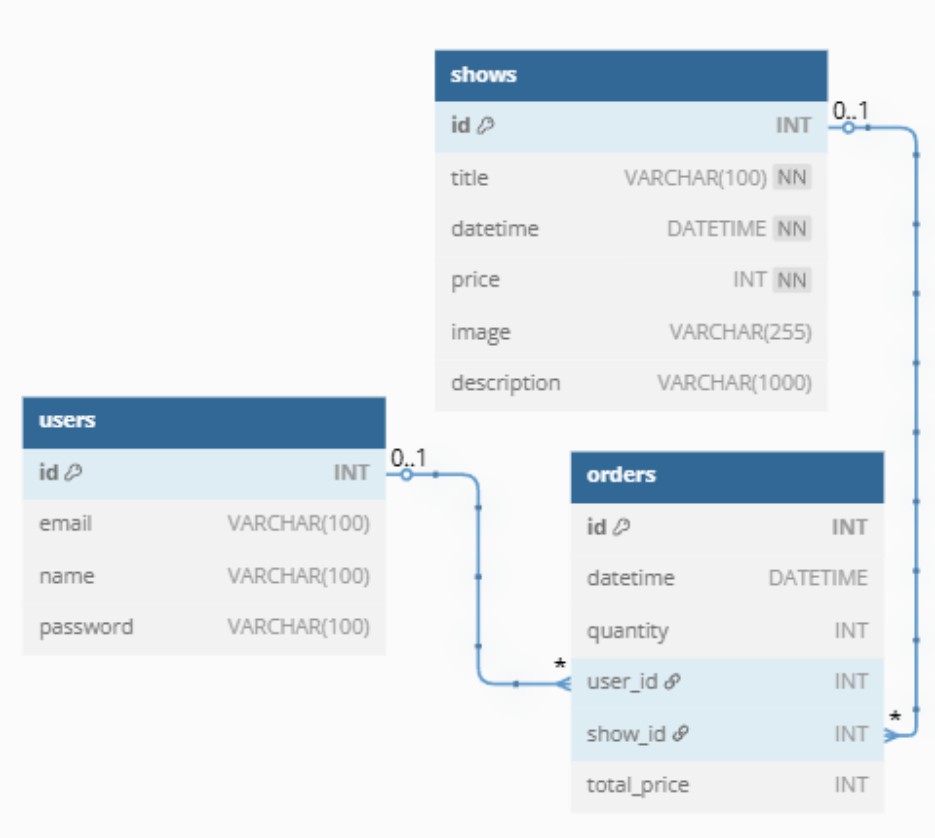
데이터베이스 모델링

주제 : 티켓 예매 사이트

총 테이블 : 공연정보(shows), 회원정보(users), 예매현황(orders)

ERD(테이블 관계도)



관계도 설명

예매현황orders 테이블 구성에서

회원정보users, 공연정보orders를 기반으로 데이터를 불러와야 하므로,

위의 ERD와 같이 연관관계를 정하였습니다.

각 테이블 만들기

1. 공연 정보 (shows)

```
MariaDB [ticketing]> DESCRIBE shows;
+-------------+--------------+------+-----+---------+----------------+
| Field       | Type         | Null | Key | Default | Extra          |
+-------------+--------------+------+-----+---------+----------------+
| id          | int(11)      | NO   | PRI | NULL    | auto_increment |
| title       | varchar(100) | NO   |     | NULL    |                |
| datetime    | datetime     | NO   |     | NULL    |                |
| price       | int(11)      | NO   |     | NULL    |                |
| image       | varchar(255) | YES  |     | NULL    |                |
| description | varchar(1000)| YES  |     | NULL    |                |
+-------------+--------------+------+-----+---------+----------------+
```

2. 회원 정보 (users)

```
MariaDB [ticketing]> CREATE TABLE users (
    -> id INT AUTO_INCREMENT PRIMARY KEY,
    ->  email VARCHAR(100),
    -> name VARCHAR(100),
    -> password VARCHAR(100)
    -> );
Query OK, 0 rows affected (0.026 sec)
```

3. 예매 현황 (orders)

```
MariaDB [ticketing]> DESCRIBE orders;
+----------+----------+------+-----+---------+----------------+
| Field    | Type     | Null | Key | Default | Extra          |
+----------+----------+------+-----+---------+----------------+
| id       | int(11)  | NO   | PRI | NULL    | auto_increment |
| datetime | datetime | YES  |     | NULL    |                |
| quantity | int(11)  | YES  |     | NULL    |                |
| user_id  | int(11)  | YES  | MUL | NULL    |                |
| show_id  | int(11)  | YES  | MUL | NULL    |                |
+----------+----------+------+-----+---------+----------------+
5 rows in set (0.001 sec)
```

2. 테이블 데이터 넣고INSERT, 조회하기SELECT

shows

```
MariaDB [ticketing]> INSERT INTO shows (title, datetime, price, image, description) VALUES
("공연1", NOW(), 50000, 'img1.jpg', '공연1 설명입니다.');
Query OK, 1 row affected (0.002 sec)

MariaDB [ticketing]> SELECT * FROM shows;
+----+---------+---------------------+-------+----------+-------------------------+
| id | title   | datetime            | price | image    | description             |
+----+---------+---------------------+-------+----------+-------------------------+
|  1 | 공연1   | 2025-04-10 09:16:34 | 50000 | img1.jpg | 공연1 설명입니다.       |
+----+---------+---------------------+-------+----------+-------------------------+
1 row in set (0.013 sec)
```

users

```
MariaDB [ticketing]> INSERT INTO users
    -> (email, name, password)
    -> VALUES('bomin@naver.com', 'bomin', 'kim1234');
Query OK, 1 row affected (0.002 sec)

MariaDB [ticketing]> SELECT * FROM users;
+----+-----------------+-------+----------+
| id | email           | name  | password |
+----+-----------------+-------+----------+
|  1 | bomin@naver.com | bomin | kim1234  |
+----+-----------------+-------+----------+
```

orders

```
MariaDB [ticketing]> INSERT INTO orders (datetime, quantity, user_id, show_id)  VALUES (NOW
(), 2, 1, 1);
Query OK, 1 row affected (0.014 sec)

MariaDB [ticketing]> SELECT * FROM orders;
+----+---------------------+----------+---------+---------+
| id | datetime            | quantity | user_id | show_id |
+----+---------------------+----------+---------+---------+
|  1 | 2025-04-10 09:23:51 |        2 |       1 |       1 |
+----+---------------------+----------+---------+---------+
```

3.users 특정 데이터 수정하기

```
MariaDB [ticketing]> UPDATE users SET password = 'bominjjang123' WHERE id = 1;
Query OK, 1 row affected (0.015 sec)
Rows matched: 1  Changed: 1  Warnings: 0

MariaDB [ticketing]> SELECT * FROM users;
+----+-----------------+-------+--------------+
| id | email           | name  | password     |
+----+-----------------+-------+--------------+
|  1 | bomin@naver.com | bomin | bominjjang123 |
+----+-----------------+-------+--------------+
```

4.실제 API만들때 필요해보이는 쿼리?, 로직?

"제한 사항"에 필요한 로직

- 한 공연당 한사람의 최대 예매 제한수는 100장

  1. 예매 완료 처리전, {주문자의 user_email과 공연의 show_title} 을 기준점으로 orders 테이블과 기준점이 겹치는 데이터(match)들을 불러온다.
  2. 그 match데이터들이 가지고있는 quantity(예매수량)을 총합하여
  3. "현재 구매하려는 수량 + 기존 총합" 이 100을 넘냐를 검사하여
  4. 넘지 않으면 예매OK / 넘으면 예매X

- orders테이블에서 외래키로 연결한 데이터에서 특정 데이터 JOIN해서 불러오기

  1. user_id -> users.name, users.email
  2. show_id -> shows.title, shows.price