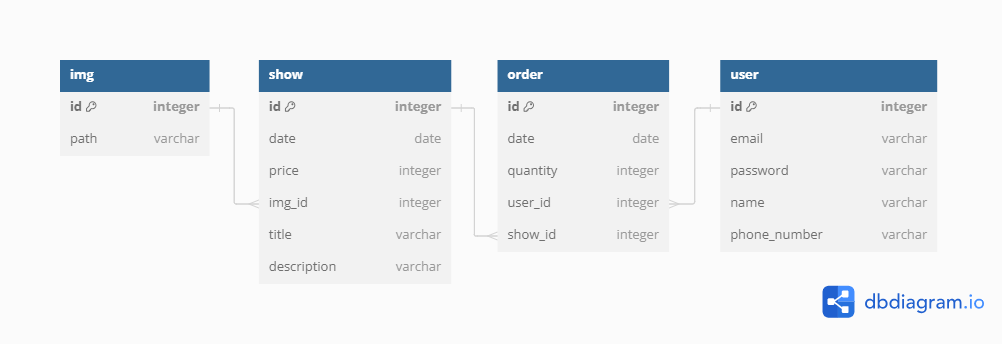
데이터베이스 모델링

b조 허대웅

# ERD



## 설명

pdf에서 나온 데이터 요구사항 이미지 순으로 설명하겠습니다.

**메인 페이지**의 경우, 공연 이미지, 날짜, 공연명, 공연 가격의 정보가 **공연 상세 페이지**에 나오는 정보와 완전히 겹치고, 나머지는 로그인 및 주문 내역 페이지로 가는 버튼이기 때문에 여기에서 설명할 내용은 없습니다.

**공연 상세 페이지**의 경우,

* 공연 이미지 - show.img\_id
* 공연명 - show.title
* 날짜 - show.date
* 공연 가격 - show.price
* 공연 소개 - show.description

으로 저장합니다.

**로그인 페이지**의 경우, 따로 데이터 베이스가 필요한 정보가 없습니다.

**회원가입 페이지**의 경우,

* 이름 - user.name
* 이메일 - user.email
* 비밀번호 - user.password

로 저장합니다.

**주문 내역 페이지**의 경우,

* 예매 번호 - order.id
* 예매 일자 - order.date
* 예매 공연명 - order.show\_id -> 해당하는 show.title
* 공연 날짜 - order.show\_id -> 해당하는 show.date
* 공연 가격 - order.show\_id -> 해당하는 show.price
* 예매자 이름 - order.user\_id -> 해당하는 user.name
* 예매자 연락처 - order.user\_id -> 해당하는 user.phone\_number
* 예매 수량 - order.quantity
* 총 예매 가격 - show.price 및 order.quantity를 통한 서버에서의 계산

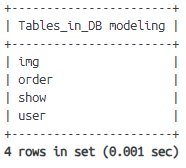
으로 저장합니다.

## 몇 가지 예상 질문

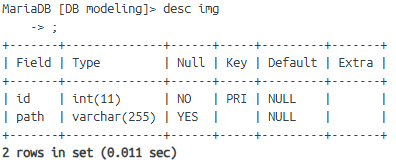
* 왜 img를 다른 테이블로 분리하였는가?  
  img에 캡션 등 **다른 속성이 추가**될 여지가 충분히 있습니다.  
  또한, img의 경로를 수정하는 등의 쿼리가 show 테이블에서 이루어지는 것이 알맞지 않다고 생각합니다.
* user 테이블에 phone\_number column이 있는 이유는?  
  회원 가입할 때에는 예매자의 연락처를 기입하지 않지만, **주문 내역에서 예매자의 연락처를 표시**하기 때문입니다.   
  다른 요구 사항이 없어 알기 어렵지만, 만약 제가 설계를 한다면 개인 정보란에서 기본 연락처를 설정하여 user.phone\_number에 저장할 것입니다.   
  다른 방법으로는, user.phone\_number column을 만들지 않고, 주문 시에 연락처를 기입, order 테이블에 연락처를 저장할 수도 있습니다. 하지만 이 경우에는 user의 정보가 order에 들어가 있기 때문에 **데이터의 일관성을 해친다**고 생각합니다.
* 왜 총 예매 가격을 저장하지 않았는가?  
  총 예매 가격은 공연 가격 및 예매 수량으로 서버에서 충분히 계산할 수 있는 데이터입니다. 이러한 데이터를 데이터 베이스에 저장하게 되면, 예매 수량 등을 수정할 때마다 데이터의 불일치를 막기 위해 총 예매 가격도 수정해야하고, 이는 DB에 쓸데없는 쿼리를 보내게 되는 것입니다.  
  이렇게 서버에서 총 예매 가격을 처리함으로써 **쿼리의 양과 데이터 저장 공간을 동시에 줄일 수 있습니다.**

# 테이블 및 예상 SQL문

## 테이블 목록



### img 테이블



#### 테이블 생성문

CREATE TABLE `img` (

`id` integer PRIMARY KEY,

`path` varchar(255)

);

#### 입력

* 이미지 입력

INSERT \* INTO `img` VALUES (1, '/img/hamlet');

#### 수정

* 경로 수정

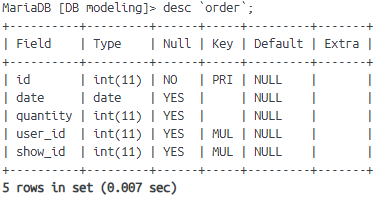
UPDATE `img` SET `path`='/src/img/hamlet' WHERE `id`=1;

#### 조회

* 경로 조회

SELECT `path` FROM `img` WHERE `id`=1;

### order 테이블



#### 테이블 생성문

CREATE TABLE `order` (

`id` integer PRIMARY KEY,

`date` date,

`quantity` integer,

`user\_id` integer,

`show\_id` integer

);

#### 입력

* 예매 주문 입력

INSERT \* INTO `order` VALUES (1, NOW(), 3, 1, 1);

#### 수정

* 주문의 수량 수정

UPDATE `order` SET `date`=NOW(), `quantity`=2 WHERE `id`=1;

수정이 될 때, 주문의 예매 날짜도 지금으로 변경합니다.

#### 조회

* 총 예매 가격을 계산하기 위한 수량 및 가격 조회

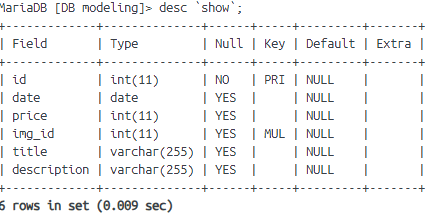
SELECT `quantity`,

(SELECT `price` FROM `show` WHERE `id`=`show\_id`) AS `price`

FROM `order` WHERE `id`=1;

수량과, show\_id가 show.id와 같은 레코드의 price를 조회합니다.

### show 테이블



#### 테이블 생성문

CREATE TABLE `show` (

`id` integer PRIMARY KEY,

`date` date,

`price` integer,

`img\_id` integer,

`title` varchar(255),

`description` varchar(255)

);

#### 입력

* 공연 입력

INSERT \* INTO `show`

VALUES (1, '2024-03-11', 10000, 1, 'hamlet',

'To be or not to be—that is the question');

#### 수정

* 공연의 가격 수정

UPDATE `show` SET `price`=12000 WHERE `id`=1;

* 공연의 제목 및 설명 수정

UPDATE `show`

SET `title`='햄릿', `description`='사느냐, 죽느냐, 그것이 문제로다'

WHERE `id`=1;

#### 조회

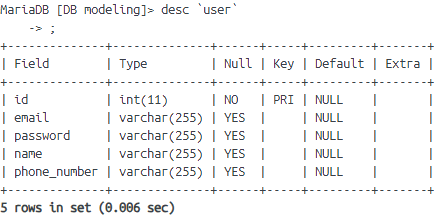
* 메인 페이지를 위한 데이터 조회

SELECT (SELECT `path` FROM `img` WHERE `id`=`img\_id`) AS `img\_path`,

`date`, `price`, `title`

WHERE `id`=1;

### user 테이블



#### 테이블 생성문

CREATE TABLE `user` (

`id` integer PRIMARY KEY,

`email` varchar(255),

`password` varchar(255),

`name` varchar(255),

`phone\_number` varchar(255)

);

#### 입력

* 유저 가입

INSERT \* INTO `show` VALUES (1, 'example@google.com', 'password1234', '홍길동', NULL);

#### 수정

* 전화 번호 수정

UPDATE `user` SET `phone\_number`='01012341234' WHERE `id`=1;

#### 조회

* 이메일에 대한 비밀번호 조회

SELECT `password` FROM `user` WHERE `email`=`example@google.com`;