

# **VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ**

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH  
TECHNOLOGIÍ

## **PROJEKT BDS**

Návrh a implementace databázové aplikace na téma:

## **WEBOVÝ PORTÁL KINA**

Autoři:

Martin Šimkovský (230679)

Martin Smetana (230668)

# OBSAH

<b>ÚVOD.....</b>	<b>2</b>
<b>ODKAZ NA GITHUB/GITLAB REPOZITÁŘ.....</b>	<b>3</b>
<b>NÁVRH DATABÁZE – ER DIAGRAM .....</b>	<b>4</b>
<b>DEFINICE JEDNOTLIVÝCH TABULEK .....</b>	<b>5</b>
1.1    TABULKA USERS .....	5
1.2    TABULKA MEMBERSHIP .....	5
1.3    TABULKA ADDRESS .....	5
1.4    TABULKA USER_HAS_ADDRESS .....	6
1.5    TABULKA CINEMA .....	6
1.6    TABULKA NEWS.....	6
1.7    TABULKA HALL .....	7
1.8    TABULKA SEAT.....	7
1.9    TABULKA SHOWS.....	7
1.10   TABULKA USER_HAS_SHOW .....	8
1.11   TABULKA MOVIE .....	8
1.12   TABULKA GENRE .....	8
1.13   TABULKA MOVIE_HAS_GENRE .....	8
<b>ZDŮVODNĚNÍ 3 NORMÁLNÍ FORMY.....</b>	<b>9</b>
<b>DLL SKRIPT NA VYTVOŘENÍ DB V POSTGRESQL.....</b>	<b>10</b>
<b>DDL SKRIPT NA VYTVOŘENÍ DB V MYSQL .....</b>	<b>13</b>
<b>PROKÁZÁNÍ VYTVOŘENÍ DATABÁZE A TABULEK .....</b>	<b>16</b>

## ÚVOD

Cílem tohoto projektu je se co nejvíce přiblížit navržení a vytvoření databáze, později uživatelského rozhraní, které by mělo fungovat jako rezervační portály multikin, které jsme zvyklí běžně využívat v dnešní době.

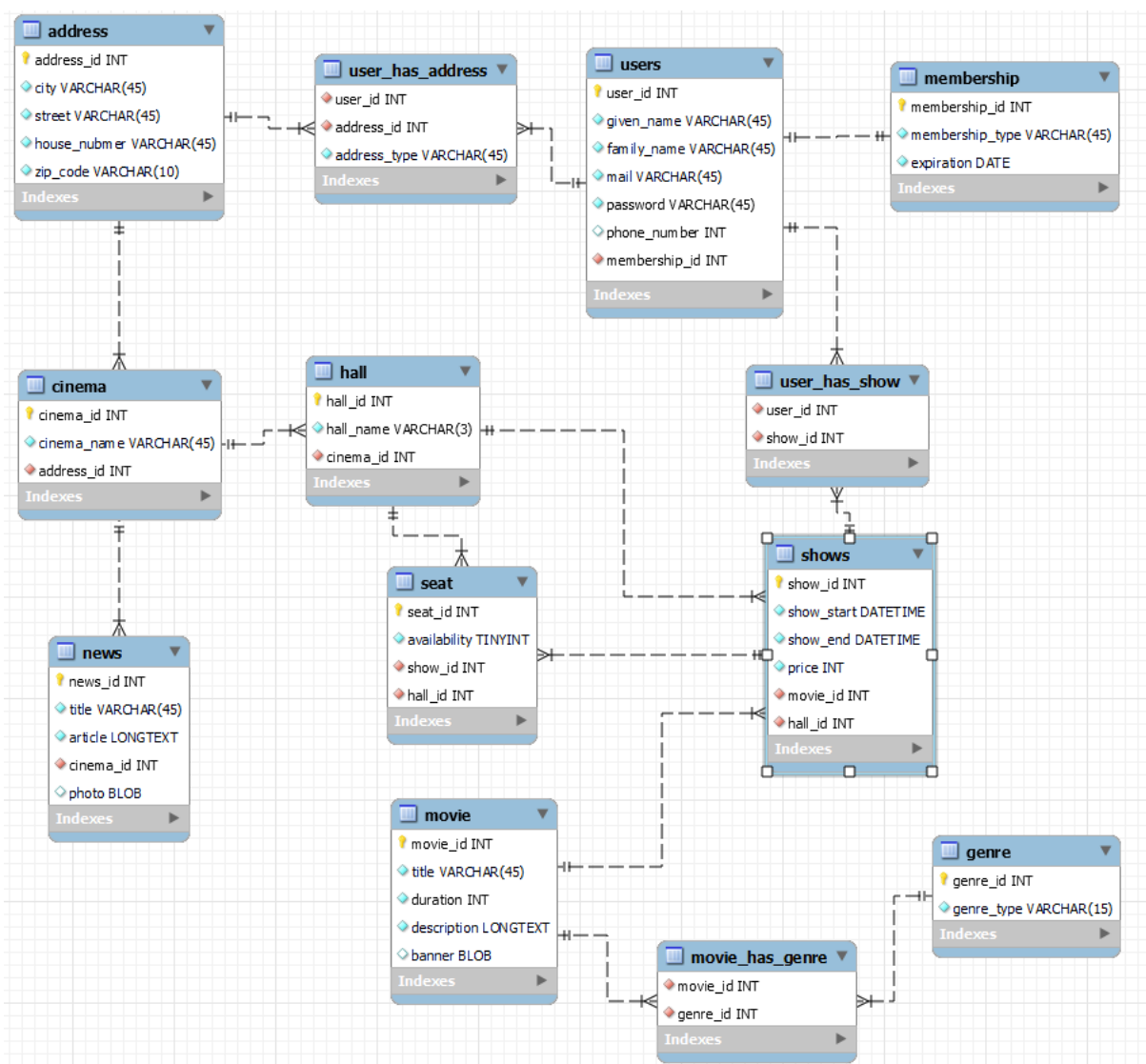
Nepřihlášený uživatel bude mít k dispozici k vylistování veškeré představení, které se budou konat. Bude také moci zobrazit obsazenost konkrétních představení(sálů). Dále zde bude možnost zobrazit aktuality daného multikina a informace o jednotlivých členstvích.

Přihlášený uživatel bude mít možnost vytvořit samotnou rezervaci na dané představení, mít správu nad svým účtem, změny členství apod.

## ODKAZ NA GITHUB/GITLAB REPOZITÁŘ

- Mrcassin/bds-db-design (github.com)
- <https://gitlab.com/xsmeta08/bds-db-design>

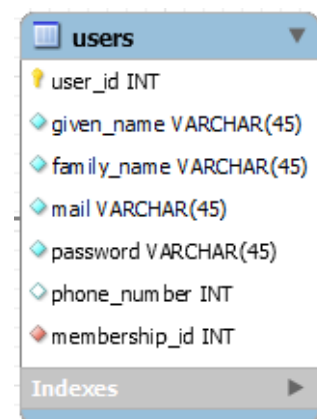
# NÁVRH DATABÁZE – ER DIAGRAM



# DEFINICE JEDNOTLIVÝCH TABULEK

## 1.1 Tabulka users

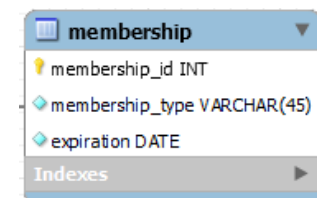
- Tabulka user je základní tabulkou pro přihlášení uživatele a jeho správě. Jsou zde zaznamenávány veškeré údaje o něm.
  - user\_id- primární klíč, datového typu serial-> automatická inkrementace
  - given\_name- krátký text -> varchar(45)
  - family\_name- krátký text -> varchar(45)
  - mail- krátký text -> varchar(45)
  - password- krátký text -> varchar(45)
  - phone\_number- číslo -> int (nepovinný atribut)
  - membership\_id- cizí klíč odkazující na členství uživatele -> integer not null



users	
user_id	INT
given_name	VARCHAR(45)
family_name	VARCHAR(45)
mail	VARCHAR(45)
password	VARCHAR(45)
phone_number	INT
membership_id	INT

## 1.2 Tabulka membership

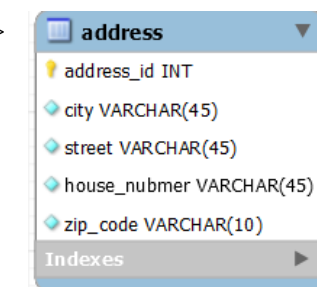
- Tabulka membership slouží k definici typu členství uživatele. K tabulce user je zde poměr 1:1, jelikož jeden uživatel nemůže mít více členství najednou.
  - membership\_id- primární klíč, datového typu serial - > automatická inkrementace
  - membership\_type- krátký text -> varchar(45)
  - expiration- datum -> DATE



membership	
membership_id	INT
membership_type	VARCHAR(45)
expiration	DATE

## 1.3 Tabulka address

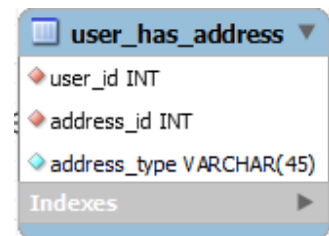
- Tabulka address je základní tabulkou pro uložení všech adres.
  - address\_id- primární klíč, datového typu serial-> automatická inkrementace
  - city- krátký text -> varchar(45)
  - street- krátký text -> varchar(45)
  - house\_number- krátký text -> varchar(45)
  - zipcode- krátký text -> varchar(10)



address	
address_id	INT
city	VARCHAR(45)
street	VARCHAR(45)
house_number	VARCHAR(45)
zipcode	VARCHAR(10)

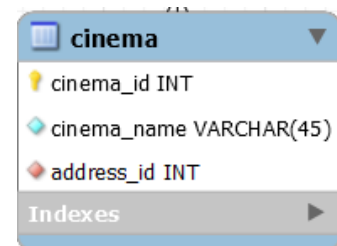
## 1.4 Tabulka user\_has\_address

- Tabulka k propojení uživatele a jeho adresy, zprostředkování vztahu mezi user a address typem N:N. Uživatel může mít více typů adres.
  - user\_id- cizí klíč odkazující na daného uživatele -> integer not null
  - Address\_id- cizí klíč odkazující na konkrétní adresu -> integer not null
  - Address\_type- krátký text -> varchar(15)



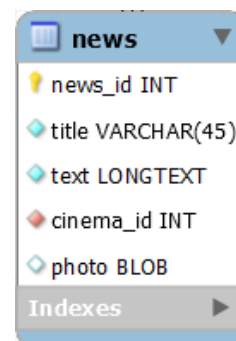
## 1.5 Tabulka cinema

- Tabulka jednotlivých kin, které se nachází na určitých adresách. 1:N
  - cinema\_id- primární klíč, datového typu serial-> automatická inkrementace
  - cinema\_name- krátký text -> varchar(45)
  - address\_id- cizí klíč odkazující na adresu na kterém se kino nachází -> integer not null



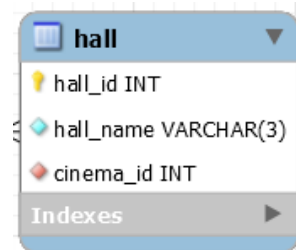
## 1.6 Tabulka news

- Tabulka aktualit daného multikina. 1:N
  - news\_id- primární klíč, datového typu seriál -> automatická inkrementace
  - title- krátký text -> varchar(45)
  - article- dlouhý text -> text
  - Photo- fotka -> text (nenašli jsme lepší alternativu bdatového typu blob v postgresql -> binární zápis obrázku do datového typu text)
  - Cinema\_id- cizí klíč odkazující na kino které aktualitu zveřejnilo -> int not null



## 1.7 Tabulka hall

- V každém kině je několik sálů. 1:N
  - hall\_id- primární klíč, datového typu serial-> automatická inkrementace
  - hall\_name- jméno sálu (možná kombinace čísla a znaku) -> varchar(3)
  - cinema\_id- cizí klíč odkazující na kino ve které se sál nachází -> integer not null



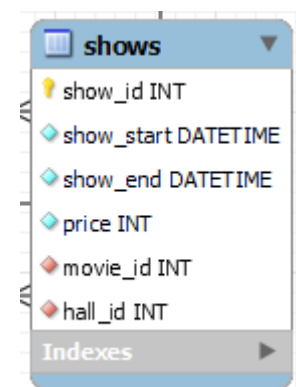
## 1.8 Tabulka seat

- V každém sále je několik sedadel. 1:N
  - seat\_id- primární klíč, datového typu serial-> automatická inkrementace
  - availability- dostupnost sedadla na daném představení -> boolean
  - show\_id- provázání k danému představení -> cizí klíč -> integer not null
  - hall\_id- provázání k danému sálu -> cizí klíč -> integer not null



## 1.9 Tabulka shows

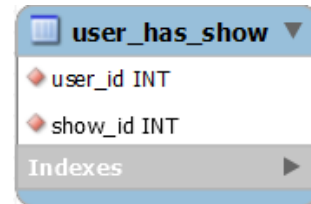
- Show je druhá stěžejní tabulka, ve které se nachází aktuální nabídka představení propojených na jednotlivé kino a sál. V jednom sále se může v jiném čase odehrávat N představení -> vztah 1:N. Jeden film může být odehrán na více představeních tedy N:1.
  - show\_id- primární klíč, datového typu seriál -> automatická inkrementace
  - show\_start -> timestamp
  - show\_end -> timestamp
  - price- číselný údaj o ceně -> integer
  - movie\_id- cizí klíč odkazující na film daného představení -> integer not null
  - hall\_id- cizí klíč odkazující na sál ve kterém bude představení probíhat -> integer not null





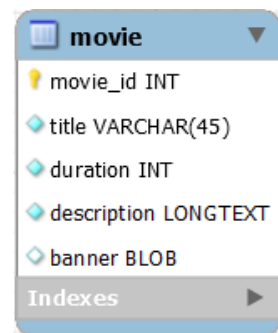
## 1.10 Tabulka user\_has\_show

- Propojovací tabulka na vytvoření N:N vztahu mezi uživatelem a představeními, kterých si může rezervovat neomezený počet.
  - user\_id- cizí klíč odkazující na konkrétního uživatele -> integer not null
  - show\_id- cizí klíč odkazující na konkrétní představení -> integer not null



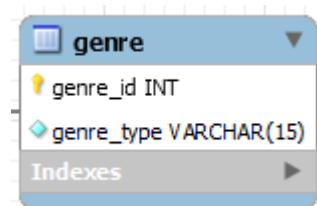
## 1.11 Tabulka movie

- Zde jsou uloženy jednotlivé filmy, které mohou být přehrávány. Jeden film může být přehrán na více představeních. 1:N.
  - movie\_id- primární klíč, datového typu seriál -> automatická inkrementace
  - title- název filmu -> krátký text -> varchar(45)
  - duration- délka filmu v minutách -> integer
  - description- delší popis filmu -> text
  - banner- obrázek -> binární zápis -> text



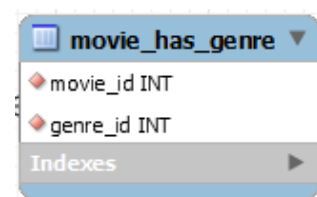
## 1.12 Tabulka genre

- Jednotlivé žánry filmů
  - genre\_id- primární klíč, datového typu seriál -> automatická inkrementace
  - genre\_type- název žánru -> krátký text varchar(15)



## 1.13 Tabulka movie\_has\_genre

- Každý film může mít více žánrů a zároveň jeden žánr může zařadit více filmů -> vztah N:N.
  - Movie\_id- cizí klíč odkazující na film -> int not null
  - Genre\_id- cizí klíč odkazující na žánr -> int not null



## ZDŮVODNĚNÍ 3 NORMÁLNÍ FORMY

Naše databáze splňuje požadavky 3. normální formy, jelikož každý atribut v každé tabulce už je dále nedělitelný neboli atomický. V databázi se rovněž nenachází žádná duplicitní data a pro každá data s jiným významem je vytvořena zvláštní tabulka. Dále splňujeme také podmínku, že každý prvek v tabulce musí být plně závislý na primárním klíči (s výjimkou města v tabulce adresa a ceny v tabulce shows), jinak se nachází v samostatné tabulce a je propojen pomocí cizího klíče.

## DLL SKRIPT NA VYTVOŘENÍ DB V POSTGRESQL

Apr 10, 2019 - Codeshare

```
create table "membership"
(
    membership_id serial primary key not null,
    membership_type varchar(45) not null,
    expiration date not null
);

create table "user"
(
    user_id serial primary key not null,
    given_name varchar(45) not null,
    family_name varchar(45) not null,
    mail varchar(45) not null,
    password varchar(45) not null,
    phone_number integer,
    membership_id integer not null,
    foreign key("membership_id") references
membership("membership_id")
);

create table "address"
(
    address_id serial primary key not null,
    city varchar(45) not null,
    street varchar(45) not null,
    house_number varchar(45) not null,
    zip_code varchar(45) not null
);

create table "user_has_address"
(
    user_id integer not null,
    address_id integer not null,
    foreign key("user_id") references "user"("user_id"),
    foreign key("address_id") references address("address_id"),
    address_type varchar(45) not null
);

create table "cinema"
(
    cinema_id serial primary key not null,
    cinema_name varchar(45) not null,
    address_id integer not null,
    foreign key("address_id") references address("address_id")
);
```

```
);

create table "hall"
(
    hall_id serial primary key not null,
    hall_name varchar(3) not null,
    cinema_id integer not null,
    foreign key("cinema_id") references cinema("cinema_id")
);

create table "genre"
(
    genre_id serial primary key not null,
    genre_type varchar(45) not null
);

create table "movie_has_genre"(
    movie_id integer not null,
    genre_id integer not null,
    foreign key ("movie_id") references movie("movie_id"),
    foreign key ("genre_id") references genre("genre_id")
);

create table "movie"
(
    movie_id serial primary key not null,
    title varchar(45) not null,
    duration integer not null,
    description text not null,
    banner text,
);

create table "show"
(
    show_id serial primary key not null,
    show_start timestamp not null,
    show_end timestamp not null,
    price integer not null,
    movie_id integer not null,
    hall_id integer not null,
    foreign key("movie_id") references movie("movie_id"),
    foreign key("hall_id") references hall("hall_id")
);

create table "user_has_show"
(
    user_id integer not null,
    show_id integer not null,
    foreign key("user_id") references "user"("user_id"),
```

```
        foreign key("show_id") references "show"("show_id")
    );

create table "news"
(
    news_id serial primary key not null,
    title varchar(45) not null,
    article text not null,
    cinema_id integer not null,
    foreign key("cinema_id") references cinema("cinema_id"),
    photo text
);

create table "seat"
(
    seat_id serial primary key not null,
    availability boolean not null,
    show_id integer not null,
    hall_id integer not null,
    foreign key("show_id") references "show"("show_id"),
    foreign key("hall_id") references "hall"("hall_id")
);
```

## DDL SKRIPT NA VYTVOŘENÍ DB V MYSQL

```
create table membership
(
    membership_id int AUTO_INCREMENT primary key not null,
    membership_type varchar(45) not null,
    expiration date not null
);

create table users
(
    user_id int AUTO_INCREMENT primary key not null,
    given_name varchar(45) not null,
    family_name varchar(45) not null,
    mail varchar(45) not null,
    password varchar(45) not null,
    phone_number integer,
    membership_id integer not null,
    foreign key(membership_id) references membership(membership_id)
);

create table address
(
    address_id int AUTO_INCREMENT primary key not null,
    city varchar(45) not null,
    street varchar(45) not null,
    house_number varchar(45) not null,
    zip_code varchar(45) not null
);

create table user_has_address
(
    user_id integer not null,
    address_id integer not null,
    foreign key(user_id) references users(user_id),
    foreign key(address_id) references address(address_id),
    address_type varchar(45) not null
);

create table cinema
(
    cinema_id int AUTO_INCREMENT primary key not null,
    cinema_name varchar(45) not null,
    address_id integer not null,
    foreign key(address_id) references address(address_id)
);

create table hall
```

```
(
    hall_id int AUTO_INCREMENT primary key not null,
    hall_name varchar(3) not null,
    cinema_id integer not null,
    foreign key(cinema_id) references cinema(cinema_id)
);

create table genre
(
    genre_id int AUTO_INCREMENT primary key not null,
    genre_type varchar(45) not null
);

create table movie
(
    movie_id int AUTO_INCREMENT primary key not null,
    title varchar(45) not null,
    duration integer not null,
    description text not null,
    banner blob
);

create table movie_has_genre(
    movie_id integer not null,
    genre_id integer not null,
    foreign key (movie_id) references movie(movie_id),
    foreign key (genre_id) references genre(genre_id)
);

create table shows
(
    show_id int AUTO_INCREMENT primary key not null,
    show_start datetime not null,
    show_end datetime not null,
    price integer not null,
    movie_id integer not null,
    hall_id integer not null,
    foreign key(movie_id) references movie(movie_id),
    foreign key(hall_id) references hall(hall_id)
);

create table user_has_show
(
    user_id integer not null,
    show_id integer not null,
    foreign key(user_id) references users(user_id),
    foreign key(show_id) references shows(show_id)
);
```

```
create table news
(
    news_id int AUTO_INCREMENT primary key not null,
    title varchar(45) not null,
    article text not null,
    cinema_id integer not null,
    foreign key(cinema_id) references cinema(cinema_id),
    photo blob
);

create table seat
(
    seat_id int AUTO_INCREMENT primary key not null,
    availability boolean not null,
    show_id integer not null,
    hall_id integer not null,
    foreign key(show_id) references shows(show_id),
    foreign key(hall_id) references hall(hall_id)
);
```



# PROKÁZÁNÍ VYTVOŘENÍ DATABÁZE A TABULEK

phpMyAdmin interface showing the 'bds\_db\_design' database structure. The 'Tables' tab is selected, displaying a list of 13 tables: address, cinema, genre, hall, membership, movie, movie\_has\_genre, news, seat, shows, users, user\_has\_address, and user\_has\_show. Each table has icons for structure, SQL, search, etc. The 'Create table' button is visible at the bottom.

pgAdmin 4 interface showing the 'bds\_db\_design/postgres@PostgreSQL 14' database structure. The 'Tables' tab is selected, displaying a list of 13 tables: address, cinema, genre, hall, membership, movie, movie\_has\_genre, news, seat, show, user, user\_has\_address, and user\_has\_show. The 'CREATE TABLE' query is visible in the Query Editor, and the 'Query returned successfully in 695 msec.' message is highlighted.

```
CREATE TABLE "user"
(
    user_id SERIAL PRIMARY KEY NOT NULL,
    given_name VARCHAR(45) NOT NULL,
    family_name VARCHAR(45) NOT NULL,
    mail VARCHAR(45) NOT NULL,
    password VARCHAR(45) NOT NULL,
    phone_number INTEGER,
    membership_id INTEGER NOT NULL,
    FOREIGN KEY ("membership_id") REFERENCES membership ("membership_id")
);

CREATE TABLE "address"
(
    address_id SERIAL PRIMARY KEY NOT NULL,
    city VARCHAR(45) NOT NULL,
    street VARCHAR(45) NOT NULL,
    house_number VARCHAR(45) NOT NULL,
    zip_code VARCHAR(45) NOT NULL
);

CREATE TABLE "user_has_address"
(
    user_id INTEGER NOT NULL,
    address_id INTEGER NOT NULL,
    FOREIGN KEY ("user_id") REFERENCES "user" ("user_id"),
    FOREIGN KEY ("address_id") REFERENCES address ("address_id"),
    address_type VARCHAR(45) NOT NULL
);
```