Übung Datenbank LF13

# Kundenanforderungen

Als Geschäftsführer von YouCinema möchte ich eine Datenbank um meine Kunden und Filme zu erfassen. In der Datenbank sollen alle Filme, die wir anbieten eingetragen sein. Mittarbeiter können weitere Filme hinzufügen. Ich möchte auch alle Kinosäle und welcher Film zu welcher Zeit dort abgespielt wird erfassen damit meine Mitarbeiter Reservierungen tätigen können. (In Zukunft möchten wir vielleicht auch eine Webseite erstellen wo die Kunden sich anmelden können und Reservierungen tätigen können). Außerdem möchte ich, dass alle Filme die ausgeliehen sind oder wurden gespeichert werden. So dass ich und meine Mittarbeiter jederzeit überprüfen können, welcher Film wann und an welchen Kunden ausgeliehen wurde. Es soll möglich sein die gesamte Datenbank als SQL Datei zu exportieren. Diese Funktion kann aber nur von Mittarbeitern durchgeführt werden. Alle diese Funktionen sollen von den C# Programm aus steuer- und abrufbar sein.

# ER-Diagram

Ein Bild, das Text, Karte enthält.

Mit sehr hoher Zuverlässigkeit generierte Beschreibung

# Relationales Schema

**yc\_customer**(*ID* : VARCHAR, email : VARCAHR, firstname : VARCAHR, lastname : VARCAHR, credit : DOUBLE PRECISION, isDisabled BOOLEAN)

**yc\_cast**(*ID* : VARCHAR, firstname : VARCHAR, lastname : VARCHAR, date\_of\_birth : DATE, IMDB\_rating : DOUBLE PRECISION)

**yc\_movie**(*ID* : VARCHAR, thumbnail : VARCHAR, description VARCHAR, m\_name : VARCHAR, publishing\_year : VARCHAR, price\_per\_day\_borrow DOUBLE PRECISION, duration : TIME)

**yc\_rooms**(*roomID* : VARCHAR, max\_col : INT, max\_row : INT, room\_name : VARCHAR)

**yc\_demonstration**(*demonstrationID* : VARCHAR, ticket\_prices : DOUBLE PRECISION, demonstration\_date : DATE, fk\_movieID : VARCHAR, fk\_roomID : VARCHAR)

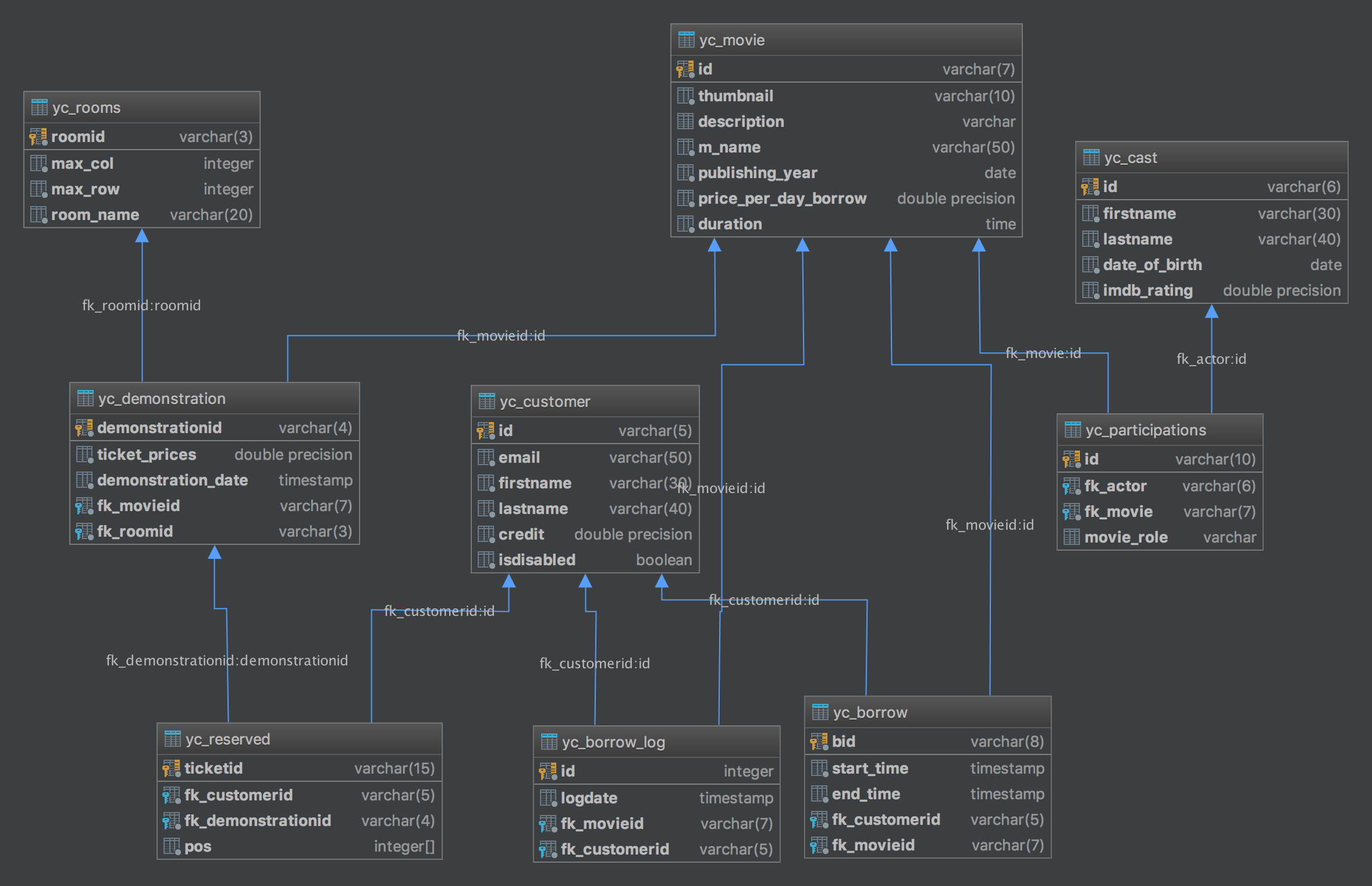
**yc\_reserved**(*ticketID* : VARCHAR, fk\_customerID : VARCHAR, fk\_demonstrationID : VARCHAR, pos : INT[])

**yc\_borrow**(*BID* : VARCHAR, start\_time : TIMESTAMP, end\_time : TIMESTAMP, fk\_customerID : VARCHAR, fk\_movieID : VARCHAR)

**yc\_borrow\_log**(*ID* : SERIAL, logdate : TIMESTAMP, fk\_movieID : VARCHAR, fk\_customerID : VARCHAR)

**yc\_participations**(*ID* : VARCHAR, fk\_actor : VARCHAR, fk\_movie : VARCHAR, movie\_role : VARCHAR)

# Object model



SQL

**!!! Bitte beachten Sie!!!**

Um die SQL ausführen zu können muss eine Datenbank bereits vorhanden sein. Wir empfählen das Sie zuerst den Folgenden Befehl ausfuhren „CREATE DATABASE youcinema;“. Nachdem dass erfolgreich war können Sie mit den SQL Code unter diesem Absatz fortfahren

-- Create ID Generator

Create or replace function random\_string(length integer) returns text as

$$

declare

chars text[] := '{0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R,S,T,U,V,W,X,Y,Z,a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,x,y,z}';

result text := '';

i integer := 0;

begin

if length < 0 then

raise exception '! < 0';

end if;

for i in 1..length loop

result := result || chars[1+random()\*(array\_length(chars, 1)-1)];

end loop;

return result;

end;

$$ language plpgsql;

-- Create Tables

CREATE TABLE yc\_customer

(

ID VARCHAR(5) DEFAULT random\_string(5) PRIMARY KEY NOT NULL,

email VARCHAR(50) NOT NULL,

firstname VARCHAR(30) NOT NULL,

lastname VARCHAR(40) NOT NULL,

credit DOUBLE PRECISION NOT NULL,

isDisabled BOOLEAN NOT NULL

);

CREATE TABLE yc\_cast

(

ID VARCHAR(6) DEFAULT random\_string(6) PRIMARY KEY NOT NULL,

firstname VARCHAR(30) NOT NULL,

lastname VARCHAR(40) NOT NULL,

date\_of\_birth DATE NOT NULL,

IMDB\_rating DOUBLE PRECISION NOT NULL

);

CREATE TABLE yc\_movie

(

ID VARCHAR(7) DEFAULT random\_string(7) PRIMARY KEY NOT NULL,

thumbnail VARCHAR(10) NOT NULL,

description VARCHAR,

m\_name VARCHAR(50) NOT NULL,

publishing\_year DATE NOT NULL,

price\_per\_day\_borrow DOUBLE PRECISION NOT NULL,

duration TIME NOT NULL

);

CREATE TABLE yc\_rooms

(

roomID VARCHAR(3) DEFAULT random\_string(3) PRIMARY KEY NOT NULL,

max\_col INT NOT NULL,

max\_row INT NOT NULL,

room\_name VARCHAR(20) NOT NULL

);

CREATE TABLE yc\_demonstration

(

demonstrationID VARCHAR(4) DEFAULT random\_string(4) PRIMARY KEY NOT NULL,

ticket\_prices DOUBLE PRECISION NOT NULL,

demonstration\_date TIMESTAMP NOT NULL,

fk\_movieID VARCHAR(7) NOT NULL,

fk\_roomID VARCHAR(3) NOT NULL,

CONSTRAINT yc\_demonstration\_yc\_movie\_id\_fk FOREIGN KEY (fk\_movieID) REFERENCES yc\_movie (id),

CONSTRAINT yc\_demonstration\_yc\_rooms\_roomid\_fk FOREIGN KEY (fk\_roomID) REFERENCES yc\_rooms (roomid)

);

CREATE TABLE yc\_reserved

(

ticketID VARCHAR(15) DEFAULT random\_string(15) PRIMARY KEY NOT NULL,

fk\_customerID VARCHAR(5) NOT NULL,

fk\_demonstrationID VARCHAR(4) NOT NULL,

pos INTEGER[] NOT NULL,

CONSTRAINT yc\_reserved\_yc\_customer\_id\_fk FOREIGN KEY (fk\_customerID) REFERENCES yc\_customer (id),

CONSTRAINT yc\_reserved\_yc\_demonstration\_demonstrationid\_fk FOREIGN KEY (fk\_demonstrationID) REFERENCES yc\_demonstration (demonstrationid)

);

CREATE TABLE yc\_borrow

(

BID VARCHAR(8) DEFAULT random\_string(8) PRIMARY KEY,

start\_time TIMESTAMP DEFAULT now() NOT NULL,

end\_time TIMESTAMP NOT NULL,

fk\_customerID VARCHAR(5) NOT NULL,

fk\_movieID VARCHAR(7) NOT NULL,

CONSTRAINT yc\_borrow\_yc\_customer\_id\_fk FOREIGN KEY (fk\_customerID) REFERENCES yc\_customer (id),

CONSTRAINT yc\_borrow\_yc\_movie\_id\_fk FOREIGN KEY (fk\_movieID) REFERENCES yc\_movie (id)

);

CREATE TABLE yc\_borrow\_log

(

ID SERIAL NOT NULL PRIMARY KEY,

logdate TIMESTAMP NOT NULL,

fk\_movieID VARCHAR(7) NOT NULL,

fk\_customerID VARCHAR(5) NOT NULL,

CONSTRAINT yc\_borrow\_log\_yc\_customer\_id\_fk FOREIGN KEY (fk\_customerID) REFERENCES yc\_customer (id),

CONSTRAINT yc\_borrow\_log\_yc\_movie\_id\_fk FOREIGN KEY (fk\_movieID) REFERENCES yc\_movie (id)

);

CREATE TABLE yc\_participations

(

id VARCHAR(10) DEFAULT random\_string(10) PRIMARY KEY NOT NULL,

fk\_actor VARCHAR(6) NOT NULL,

fk\_movie VARCHAR(7) NOT NULL,

movie\_role VARCHAR,

CONSTRAINT yc\_participations\_yc\_cast\_id\_fk FOREIGN KEY (fk\_actor) REFERENCES yc\_cast (id),

CONSTRAINT yc\_participations\_yc\_movie\_id\_fk FOREIGN KEY (fk\_movie) REFERENCES yc\_movie (id)

);

-- Create Auto Log Function

Create or replace function log\_borrow()

returns trigger as

$$

begin

INSERT INTO yc\_borrow\_log(ID, logdate, fk\_movieid, fk\_customerid)

VALUES (DEFAULT, now(), NEW.fk\_movieid, NEW.fk\_customerid);

return NEW;

end;

$$

LANGUAGE plpgsql;

-- Add Trigger to Function

CREATE TRIGGER logging

AFTER INSERT

ON yc\_borrow

FOR EACH ROW

EXECUTE PROCEDURE log\_borrow();

Abfragen:

Alle Filme kriegen:

SELECT id,thumbnail,description,m\_name,publishing\_year,price\_per\_day\_borrow,duration FROM yc\_movie

Alle Vorführungen und dessen Reservierungen löschen die schon abgelaufen sind:

DELETE FROM yc\_reserved

WHERE fk\_demonstrationid

IN

(

SELECT demonstrationid

FROM yc\_demonstration d, yc\_movie

WHERE m.id = d.fk\_movieid

AND d.demonstration\_date < now() - m.duration

);

DELETE FROM yc\_demonstration d

USING yc\_movie m

WHERE m.id = d.fk\_movieid

AND d.demonstration\_date < now() - m.duration;

Loags laden:

SELECT logdate,fk\_customerid,fk\_movieid FROM yc\_borrow\_log

TIP: Es lieg auch eine SQL Datei bei.

Server Informationen

Für den Fall das Sie YouCinema Testen wollen ohne Datenbank und MediaServer aufzusetzen. Finden Sie hier einen Demo Server der Demo material besitzt.

Achtung! Server verfügbar nur bis zum 28.05.2018

Datenbank Server: 134.255.233.233

Datenbank User: Brunner

Datenbank Password: kYqWLZB4@9mI4mT1K

Datenbank Name: youcinema\_demo

(Connection string: User ID=Brunner;Password= kYqWLZB4@9mI4mT1K;Host=134.255.233.233;Port=5432;Database=youcinema\_demo)

MediaServer: <https://web.bitnt.net/server/>

Wir wünschen Ihnen viel Spaß beim Ausprobieren.