小组名称：**POI**

订票系统

——数据库建模

项目组成员：

邓杰友 陈颂熙

吴佳琪 彭志锋

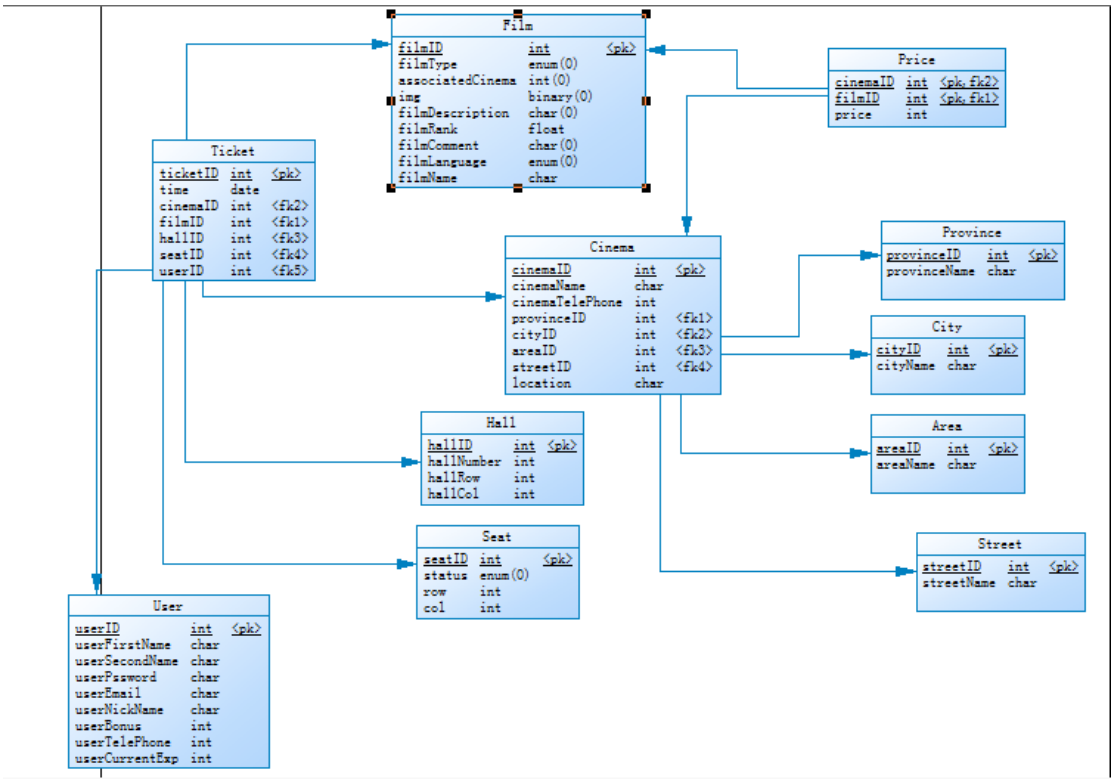
郑嘉俊 张树盛

2017 年 5 月 07 日

版本	日期	描述	作者	审阅者
V1.0	2017.5.07	数据库建模	郑嘉俊	彭志峰

数据库建模

PowerDesign 中的建模：



在整个订票系统中，一共设计了 11 个表格。它们分别是票据表（TicketTable）,用户表（UserTable）,电影表（FilmTable）,影院表（CinemaTable）,放映厅表（HallTable）,座位表（SeatTable）,价格表（PriceTable）,省份表（ProvinceTable）,城市表（CityTable）,区域表（AreaTable）,街道表（StreetTable）。

票据表，这是主要的表格，通过这个基本可以访问与这个票据有关的数据，包括这个票据的订购者（userID），电影的影院（CinemaID）,所观看的电影（FilmID），对应的放映厅（hallID），对应的座位（SeatID）。

用户表,主要用来记录用户的相关信息，包括用户的 ID，这是不可重复具有唯一性的，除此以外还包含用户的姓（userFirstName）,名（userSecondName）,用户的密码（userPassword），用户的昵称（userNickName），用户的集点（userBonus）,用户集点是每成功观看一场电影即会增加，当积攒到一定的数目可以兑换相应的奖品。

电影表。主要用来记录电影的信息，包括电影的唯一属性 filmID，电影名（filmName）,电影的类型(filmType),会播放这个电影的影院（associcatedCinema），电影的介绍图片（img）,电影的文字介绍（filmDescription），电影的评分(filmRank),电影的评价（filmComment）,电影的

语种(filmLanguage)

影院表这是用来标注影院的具体信息，包括影院的唯一属性 cinemaID,影院名字(cinemaName),影院的联系电话（cinemaTelePhone）,影院所在的省份（provinceID）,影院所在的市级单位（cityID）,影院所在的区级单位（areaID）,影院所在的街道(streetID)，街道单位以下的具体位置。（location）。

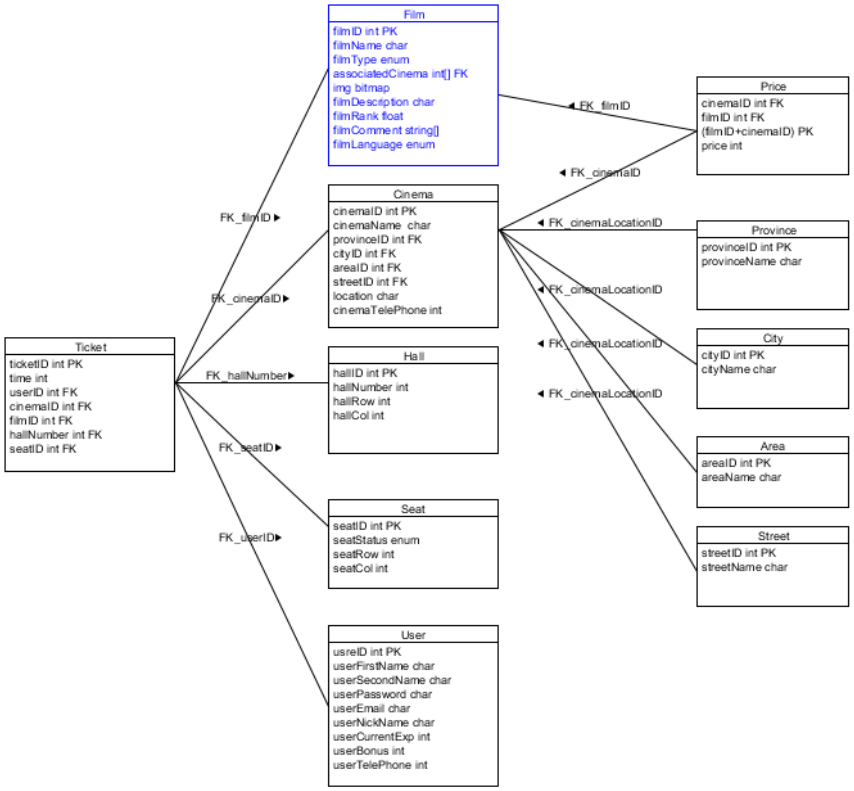
放映厅表，放映厅的唯一属性 hallID,放映厅的编号 hallNumber,这里 hallNumber 和 hallID 是不一样的，hallNumber 表示每个影院的几号厅，一般不会很大，而 hallID 则是对全国的放映厅进行一个编号。hallRow 表示这个厅有几行座位，hallCol 则表示有几列座位，获取到这些信息后就可以在客户端中把选择位置的界面显示出来。

座位表，seatID 是其字段的唯一属性，status 表示座位的当前状态，有可选，不可选，被订购，seatRow 表示座位所在行号，seatCol 表示座位所在列号。

价格表，价格表中，cinemaID 和 filmID 这两个作为这个表的复合主键，price 则是 A 影院 B 电影的价格。

省份表，城市表，地区表和街道表都是相似的，有一个唯一字段，而另外一个字段则标识该地方的名字。

以下为 UML 建模：



对应的 SQL5.0 语言

```
/*==============================================================*/
/* DBMS name:          MySQL 5.0                                */
/* Created on:         2017/5/7 星期日 18:46:50                  */
/*==============================================================*/


drop table if exists Area;

drop table if exists Cinema;

drop table if exists City;

drop table if exists Film;

drop table if exists Hall;

drop table if exists Price;

drop table if exists Province;

drop table if exists Seat;

drop table if exists Street;

drop table if exists Ticket;

drop table if exists User;

/*==============================================================*/
/* Table: Area                                                  */
/*==============================================================*/
create table Area
(
   areaID               int not null,
   areaName             char,
   primary key (areaID)
);

/*==============================================================*/
/* Table: Cinema                                                */
/*==============================================================*/
create table Cinema
```

```
(
    cinemaID                int not null,
    cinemaName                char,
    cinemaTelePhone         int,
    provinceID              int,
    cityID                  int,
    areaID                  int,
    streetID                int,
    location                char,
    primary key (cinemaID)
);

/*==============================================================*/
/* Table: City                                                  */
/*==============================================================*/
create table City
(
    cityID                  int not null,
    cityName                  char,
    primary key (cityID)
);

/*==============================================================*/
/* Table: Film                                                  */
/*==============================================================*/
create table Film
(
    filmID                  int not null,
    filmType                enum(0),
    associatedCinema        int(0),
    img                       binary(0),
    filmDescription         char(0),
    filmRank                float,
    filmComment               char(0),
    filmLanguage            enum(0),
    filmName                  char,
    primary key (filmID)
);

/*==============================================================*/
/* Table: Hall                                                  */
/*==============================================================*/
create table Hall
(
```

```
    hallID                  int not null,
    hallNumber               int,
    row                      int,
    col                     int,
    primary key (hallID)
);

/*==============================================================*/
/* Table: Price                                                 */
/*==============================================================*/
create table Price
(
    cinemaID                 int not null,
    filmID                  int not null,
    price                   int,
    primary key (cinemaID, filmID)
);

/*==============================================================*/
/* Table: Province                                              */
/*==============================================================*/
create table Province
(
    provinceID              int not null,
    provinceName             char,
    primary key (provinceID)
);

/*==============================================================*/
/* Table: Seat                                                  */
/*==============================================================*/
create table Seat
(
    seatID                  int not null,
    status                  enum(0),
    row                      int,
    col                     int,
    primary key (seatID)
);

/*==============================================================*/
/* Table: Street                                                */
/*==============================================================*/
create table Street
```

```
(
    streetID                int not null,
    streetName               char,
    primary key (streetID)
);

/*==============================================================*/
/* Table: Ticket                                                */
/*==============================================================*/
create table Ticket
(
    ticketID                int not null,
    time                     date,
    cinemaID                 int,
    filmID                  int,
    hallID                  int,
    seatID                   int,
    userID                   int,
    primary key (ticketID)
);

/*==============================================================*/
/* Table: User                                                  */
/*==============================================================*/
create table User
(
    userID                   int not null,
    userFirstName           char,
    userSecondName           char,
    userPssword             char,
    userEmail               char,
    userNickName             char,
    userBonus                int,
    userTelePhone           int,
    userCurrentExp          int,
    primary key (userID)
);

alter table Cinema add constraint FK_Reference_13 foreign key (provinceID)
        references Province (provinceID) on delete restrict on update restrict;

alter table Cinema add constraint FK_Reference_14 foreign key (cityID)
        references City (cityID) on delete restrict on update restrict;
```

```
alter table Cinema add constraint FK_Reference_15 foreign key (areaID)
        references Area (areaID) on delete restrict on update restrict;

alter table Cinema add constraint FK_Reference_16 foreign key (streetID)
        references Street (streetID) on delete restrict on update restrict;

alter table Price add constraint FK_Reference_7 foreign key (filmID)
        references Film (filmID) on delete restrict on update restrict;

alter table Price add constraint FK_Reference_8 foreign key (cinemaID)
        references Cinema (cinemaID) on delete restrict on update restrict;

alter table Ticket add constraint FK_Reference_1 foreign key (filmID)
        references Film (filmID) on delete restrict on update restrict;

alter table Ticket add constraint FK_Reference_2 foreign key (cinemaID)
        references Cinema (cinemaID) on delete restrict on update restrict;

alter table Ticket add constraint FK_Reference_3 foreign key (hallID)
        references Hall (hallID) on delete restrict on update restrict;

alter table Ticket add constraint FK_Reference_5 foreign key (seatID)
        references Seat (seatID) on delete restrict on update restrict;

alter table Ticket add constraint FK_Reference_6 foreign key (userID)
        references User (userID) on delete restrict on update restrict;
```